UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Treball Final de Grau

Renderització i interacció mitjançant Ray Marching

Memòria

Autor: Roger Nogué

Data de defensa:

Director i departament: Antonio Chica, Ciències de la Computació

Titulació: Grau en Enginyeria Informàtica

Especialitat: Computació

Centre i Universitat: FIB, UPC

Abstract

Català

En aquest document es troba tota la informació necessària per entendre completament el treball adjunt. Inicialment s'introdueix el treball i es resumeix sobre l'actualitat del mercat de productes amb característiques similars. Més endavant es fa referència a l'actualitat en àmbit d'estudis i s'esmentaran els avantatges del treball. A continuació es detalla la metodologia de desenvolupament detallant les propietats de cada pas. I finalment es detalla el contingut i les capacitats del treball amb els possibles errors que puguin sorgir.

Castellano

En el siguiente documento se encuentra toda la información necesaria para entender completamente el trabajo adjunto. Inicialmente se introduce el trabajo y se resume sobre la actualidad del mercado de productos con características similares. Más adelante se hará referencia a la actualidad en ámbito de estudios y se mencionaran las ventajas del trabajo. A continuación se detalla la metodología de desarrollo detallando las propiedades de cada paso. I finalmente se detalla el contenido y las capacidades del trabajo con los posibles errores que puedan aparecer.

English

The following document contains all the information needed in order to completely understand the attached project. First, the project is introduced and the current state of the market of similar products will be plotted. Second, it makes reference to the actuality in terms of research and mentions the advantages of the project. Third, there is an explanation of the development methodology detailing the properties of each development step. And last, the content and the possible errors that can appear during the development.

Índex de continguts

1 Introducció

1.1 Contextualització

Avui dia és molt habitual l'ús de programes d'animació 3D tant pel desenvolupament de pel·lícules, curtmetratges, videojocs, simulacions realistes... I cadascun dels àmbits on s'utilitzen aquests programes té molt mercat. Principalment les pel·lícules i els videojocs generen centenars de milions de dòlars a les empreses més punteres cada cop que publiquen un producte.

En aquest projecte es desenvoluparà un motor gràfic que permetrà a l'usuari dissenyar i reproduir escenes realistes completament capaces de produir un curtmetratge, un videojoc o una simulació de qualsevol tipus en 3D. Un cop acabat el producte, qualsevol usuari sense necessitat de gaire formació podrà utilitzar el programa per a produir qualsevol dels productes esmentats anteriorment.

Una referència que demostra que el mercat per al producte està creixent és el mateix creixement de la GDC[1] (on es reuneix una part dels potencials clients del meu producte) i l'augment de preus per assistir-hi. Actualment el preu de l'entrada que inclou accés a les conferències de realitat virtual (tipus de producte que el meu projecte permetrà desenvolupar) es troba al voltant dels 2.500 dòlars.

El projecte va principalment dirigit als desenvolupadors de les empreses de qualsevol dels sectors mencionats anteriorment. Aquests desenvolupadors actualment ja utilitzen programes similars al que s'ha desenvolupat en el projecte. Però amb els avantatges que ofereix aquest projecte, la intenció és que algunes empreses es decantin per a utilitzar el programa desenvolupat en comptes de la competència.

El principal motiu pel qual les empreses utilitzaran el projecte és perquè aquest permetrà generar les escenes de manera més eficient que la competència. Això donarà lloc a la producció d'escenes amb més elements i més detalls. Per tant, l'empresa oferirà al públic un producte de millor qualitat.

1.2 Formulació del problema

El treball té l'objectiu principal de tenir potencial per accedir al mercat i poder competir amb els programes de la competència en eficiència, en usabilitat, en possibilitats i en qualitat de resultats.Per tal d'aconseguir aquest objectiu principal, es podria separar aquest objectiu en diverses parts:

- Codi eficient: per a aconseguir l'objectiu que les escenes es poden representar en màquines no molt potents, cal aconseguir que el codi sigui eficient. Per això s'ha utilitzat un algorisme i una forma de representar les figures que s'aprofiten d'algunes característiques de les targetes gràfiques i de la paral·lelització que utilitza OpenGL a l'hora de tractar el fragment shader.

- Realisme: per a aconseguir la sensació de realisme, ha calgut implementar un algorisme i diverses ampliacions amb aquesta fi. Sempre tenint en compte que no volem perdre l'eficiència.

- Sensació de temps real: aconseguir aquesta sensació és important per a crear escenes realment realistes. No es vol donar la sensació que el motor gràfic genera imatges. Per això s'ha implementat una forma de navegar dins l'escena.

Els objectius esmentats juntament amb el raonament que es fa en la secció "estat de l'art" sobre les tecnologies actuals, els avantatges i les possibilitats del treball, permetran desenvolupar un producte competitiu amb capacitat d'accedir a diversos mercats diferents.

1.3 Abast

El projecte utilitza llibreries d'OpenGL[17] que permeten generar gràfics en 2 i 3 dimensions. El primer pas és configurar les llibreries i fer funcionar el procés tant per a Windows com per a Linux.

Un cop el programa funciona en ambdues plataformes, cal implementar l'algorisme bàsic que permeti representar escenes bàsiques de tal manera que es puguin detectar els errors comesos fins ara i procedir a arreglar-los (veure Figura 1).

Amb la representació d'escenes bàsiques, la base del projecte està implementada. A continuació s'han anat afegint ampliacions al projecte per tal de millorar la qualitat de les escenes i la interacció amb la càmera i l'escena.

Hi ha moltes possibilitats a l'hora d'implementar un motor gràfic i molts detalls possibles que poden influir molt positivament en les escenes finals. Per a resumir les implementacions, s'ha decidit dividir les ampliacions del projecte en 2 grups: ampliació de l'algorisme i ampliació de la usabilitat.

L'ampliació de l'algorisme consisteix bàsicament en el tipus de millores que formen part del procés de visualització. Per exemple, millorar el tractament de llum, afegir possibles figures a dibuixar, afegir reflexions de llum... Aquestes ampliacions no tenen per què ser trivials, però el procés de realitzar aquestes implementacions és progressiu i sempre s'avança.

L'ampliació de la usabilitat consisteix en les millores externes a l'algorisme. Ja pot ser afegir una llibreria que permeti navegar més fàcilment per l'escena, fer el projecte compatible amb més sistemes operatius o afegir interacció amb altres programes que permetin generar altres tipus de resultats que puguin ser interessants en algunes escenes. Aquest tipus d'ampliacions es poden complicar, ja que hi ha moltes variables: versió de les aplicacions, compatibilitats, procediments... I el procés d'afegir una d'aquestes funcionalitats no té per què ser senzill.

Un cop cadascuna de les eines sembla estar implementada, cal realitzar seguits de proves per assegurar el correcte funcionament i d'aquesta i assegurar que no hi ha errors que no permetin avançar. Per a veure la competitivitat de cada implementació també es pot comparar el resultat amb els productes professionals que hi ha al mercat i analitzar les diferències.

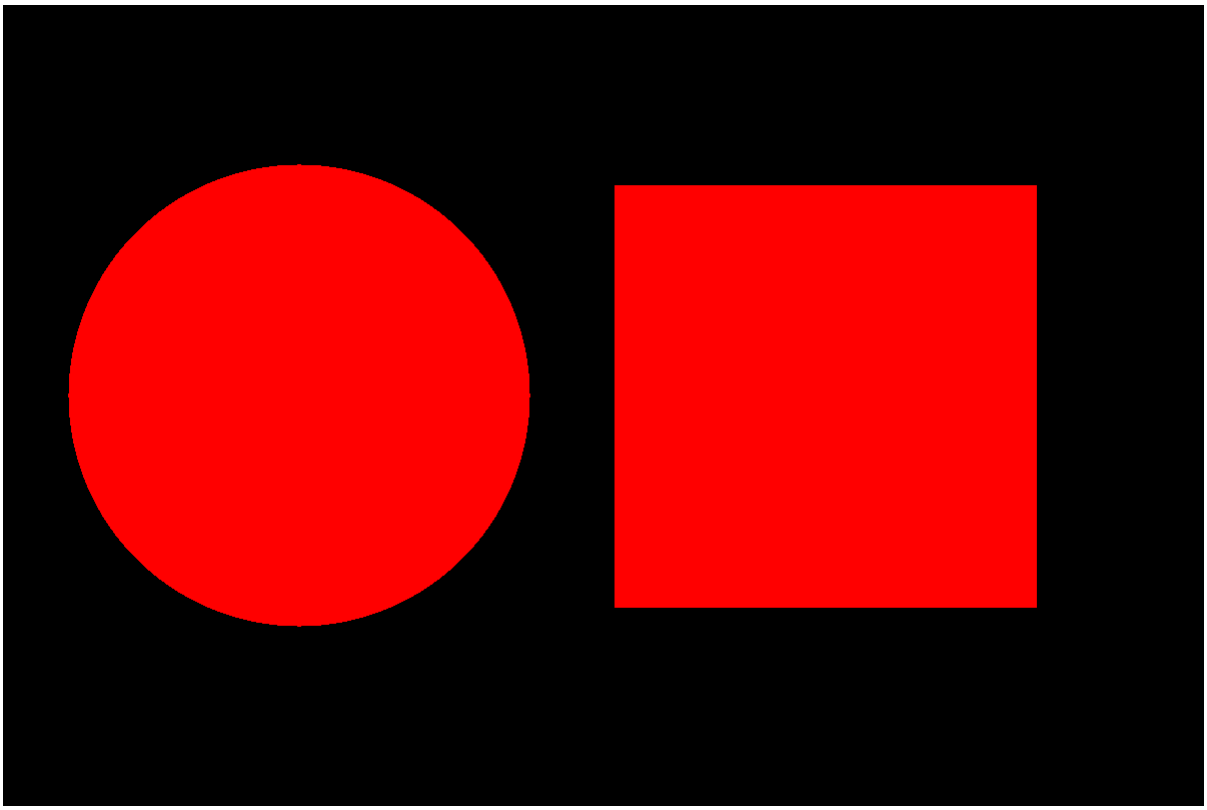


Figura 1: representació de l'escena més bàsica. Si l'algorisme detecta un objecte, el dibuixa de color vermell. I pinta negre a on no el detecta.

1.3.1 Ampliacions de l'algorisme

Com s'ha esmentat anteriorment, un cop es té la base configurada i els fonaments de l'algorisme estan implementats i funcionen correctament, les millores de l'algorisme de visualització ja es poden començar a implementar. A continuació es pot començar a treballar en el tractament de les llums, en l'augment del ventall de possibles figures a representar o en altres detalls que poden millorar la qualitat dels resultats.

El tractament de llums és un procés interessant en el projecte, ja que afegeix molt realisme si està ben implementat: il·luminació, ombres, reflexió de llum entre objectes, refracció en objectes transparents... Però que aquestes funcionalitats van de la mà amb l'augment del cost en temps d'execució de l'algorisme, així que s'haurà d'anar amb compte a l'hora de decidir quins implementar i com fer-ho.

A l'hora de considerar el tractament de noves figures en les escenes, una eina molt interessant és la intersecció, la unió i la diferència de figures geomètriques bàsiques (vegeu Figura 2). Aquesta eina permet crear molta varietat de noves figures molt interessants que milloraran la qualitat de les possibles escenes a representar.

Però les possibilitats no s'acaben aquí. Hi ha molts detalls que poden augmentar la qualitat de les escenes i/o reduir el cost de la renderització d'aquestes. Un exemple de la millora de la qualitat podria ser la implementació d'un procediment d'antialiasing, que consisteix a millorar la definició de les arestes dels objectes donant una sensació de millor qualitat. Un altre exemple podria consistir a tractar de manera diferent els objectes segons la distancia a la càmera. De tal manera que els objectes més propers es tracten amb més cura que els que es troben lluny. Això permetria reduir el cost de l'algorisme i les escenes es generarien de manera més eficient.

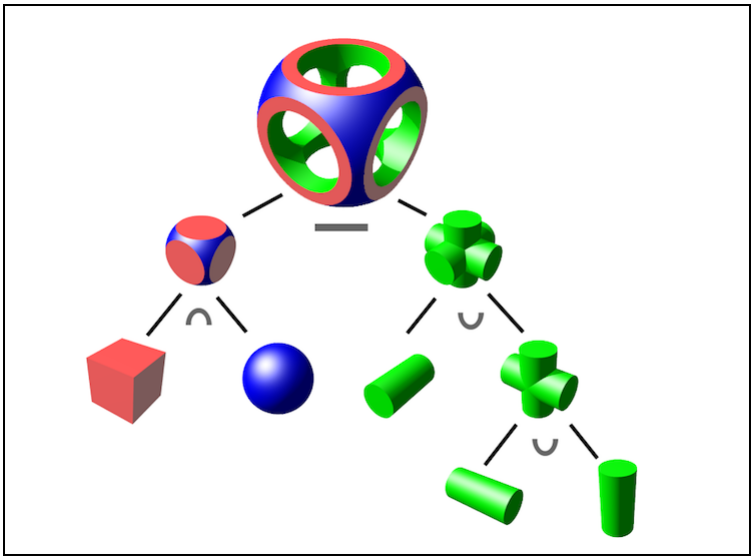


Figura 2: exemple d'una figura generada a partir de la unió, intersecció i diferència d'altres figures bàsiques.

1.3.2 Ampliacions de la usabilitat

A l'hora d'implementar aquest tipus de funcionalitats per primer cop, els problemes estan pràcticament garantits. Com s'ha esmentat anteriorment, hi ha moltes variables que poden generar errors i problemes de compatibilitat si el procediment no és el correcte o si les versions no són compatibles. La primera implementació a realitzar consisteix a fer el projecte compatible amb Linux, una altra implementació interessant és afegir el moviment de la càmera permetent a l'usuari controlar aquesta mitjançant el teclat o el ratolí, separar el codi de l'algorisme per permetre a l'usuari definir l'escena en un fitxer separat de l'algorisme i els procediments que no ha de modificar.

Per a aconseguir la compatibilitat amb Linux, cal instal·lar les llibreries adients amb la seva versió corresponent, localitzar aquestes llibreries i compilar el projecte de la forma adequada.

Per poder permetre a l'usuari el control sobre la càmera, una opció és utilitzar QT[18] que permetria afegir una interfície a l'escena. Afegir aquesta eina, ferla funcionar tant en la versió de Windows com la de Linux i implementar el moviment de la càmera pot convertir-se en un procés complex.

L'algorisme i la definició de l'escena es troben en el mateix fitxer (fragment shader). El fragment shader és un fitxer que no està dissenyat per a estar separat en diferents parts. Així que separar aquest fitxer en 2 parts (algorisme per una banda i definició d'escena per una altra) pot convertir-se en un procés complicat.

1.3.3 Possibles obstacles

Com s'ha esmentat anteriorment, hi ha un tipus d'implementacions que tendeixen a oferir més dificultats que la resta. Es té en compte que cadascuna de les ampliacions de la usabilitat té possibilitats d'oferir dificultats i problemes de compatibilitat, mentre que les dificultats que ofereixen les ampliacions de l'algorisme tendeixen a ser més fàcils de detectar i de solucionar.

2 Estat de l'art

Actualment hi ha molts procediments i algorismes diferents per a generar escenes realistes. Però després de considerar la situació actual es va decidir utilitzar l'algorisme que més es podia aprofitar d'aquesta situació i de la que es preveu en un futur pròxim. A continuació s'esmenten, es raonen i es donen referències d'estudis sobre aquests avantatges que pot presentar l'algorisme.

L'algorisme que ens proporciona els avantatges i, per tant, s'ha utilitzat en el treball s'anomena "sphere marching" o "ray marching" i presenta uns avantatges molt interessants que poden fer el treball una mica més eficient que la competència en diverses situacions. En aquests darrers anys les targetes gràfiques (GPUs) estan evolucionant més de pressa que els processadors (CPUs), i aquest algorisme s'aprofita d'aquesta situació centrant el treball en la GPU.

Com es pot veure en l'estudi[6] d'una associació que estudia l'àmbit dels gràfics, es pot aprofitar la situació actual i es plantegen diverses millores per tal de fer funcionar l'algorisme de manera menys costosa i ràpida. L'estudi esmentat és la principal font en la qual m'he basat a l'hora d'analitzar i optimitzar les propietats de l'algorisme. A partir de les referències del mateix estudi es pot accedir a altres estudis prou interessants que donen suport a la idea. També es pot veure que les dates dels estudis referenciats en el mateix estudi són més aviat recents (15 anys enrere com a molt). Hi ha algunes referències més antigues, però aquestes tracten aspectes generals sobre els gràfics i la visualització que no formen part de la identitat de l'algorisme (il·luminació, càlcul de distàncies, geometria i càlcul).

Recentment (18-22 de març d'aquest 2018) NVIDIA[9] va fer una conferència[8] on resumia i demostrava els últims avanços en els quals han estat treballant. En aquesta conferència NVIDIA posa molt d'èmfasi en la implementació de millores de rendiment i d'eficiència pels algorismes de traçat de rajos (l'algorisme utilitzat en el projecte utilitza traçat de rajos). Això dóna suport a la idea que aquest tipus d'algorismes actualment són els que més s'estan utilitzant en el mercat professional i així es preveu per als futurs anys.

NVIDIA és actualment l'empresa de fabricació de targetes gràfiques amb més èxit. Tenint en compte això, les conclusions a les quals arriben no només verifiquen que la decisió de seleccionar un algorisme de traçat de rajos era bona, sinó que el hardware que sortirà al mercat en els anys vinents també estarà preparat específicament per a millorar el funcionament d'aquest tipus d'algorismes.

3 Procés de desenvolupament

S'ha disposat d'aproximadament 4 mesos per a desenvolupar el projecte. I un projecte d'aquestes característiques té diferents fases i etapes que s'han de finalitzar per tal de poder tenir un producte prou bo. A continuació hi ha una explicació de cadascun dels processos de desenvolupament pels que ha passat el projecte.

3.1 Configuració llibreries

Abans de començar amb la implementació de les parts més tècniques del projecte cal tenir un entorn de treball adequat per tal de poder anar avançant sense tenir la necessitat de tornar enrere a mig projecte perquè alguna compatibilitat, llibreria o eina encara no està inclosa o configurada. Per aquest motiu inicialment s'ha optat per treballar en el procés d'instal·lar i configurar programes i llibreries. A continuació s'explica el procés d'instal·lació i configuració de cadascuna de les eines.

3.1.1 Microsoft Visual Studio

Aquest programa en principi no és gratuït, però per ser estudiant de la UPC disposo d'una clau amb la qual puc utilitzar-lo de manera gratuïta. Per al projecte s'ha utilitzat la versió 2017 d'aquest programa. Un cop el programa està descarregat i instal·lat, ja està l'entorn bastant ben configurat per a crear un nou projecte amb pràcticament tot el necessari per a començar el projecte. Per a poder començar a representar escenes només faltaria la inclusió de les llibreries d'OpenGL.

3.1.2 OpenGL

Amb l'entorn de desenvolupament preparat, s'han descarregat les llibreries "glew" i "freeglut" i s'han inclòs al projecte. El procés d'incloure llibreries a Microsoft Visual Studio té diverses fases i no és massa intuïtiu.

- Afegir documents .h: per afegir aquests documents cal accedir a les opcions del projecte i a l'apartat C/C++ hi ha una opció anomenada "Directorios de inclusión adicionales" on cal afegir la ruta cap als arxius .h.

- Afegir documents .lib: per a afegir aquests documents al projecte cal accedir també a les propietats del projecte i a l'apartat vinculador hi ha una opció anomenada "Directorios de bibliotecas adicionales" on cal afegir la ruta als arxius .lib.

- Afegir documents .dll: per incloure aquests documents basta amb posar l'arxiu .dll a la carpeta del projecte.

També s'ha afegit les definicions d'algunes estructures de dades que permeten utilitzar les funcionalitats d'OpenGL de forma més senzilla. Algunes d'aquestes estructures de dades són "matrix4x4", "matrix3x3", "vector3" i algunes funcions útils com el "readFile()" que facilita el procés de lectura dels shaders.

Amb aquestes inclusions realitzades el projecte ja està preparat per a cridar a la pipeline d'OpenGL i utilitzar les seves funcionalitats.

3.1.3 GitHub

Per garantir que tots els arxius es mantenen al núvol de forma segura en cas que per qualsevol motiu el terminal principal on s'està desenvolupant el projecte deixi de funcionar, s'ha utilitzat la plataforma de GitHub. Aquesta plataforma ofereix un control de versions de forma gratuïta, i això ha sigut extremadament útil durant el projecte.

En aquest projecte s'han utilitzat 3 repositoris d'aquesta plataforma:

- Projecte OpenGL Windows: Aquest repositori ha emmagatzemat principalment un projecte de Microsoft Visual Studio 2017 amb les llibreries d'OpenGL incloses, un arxiu "main.cpp" on es cridaven totes les funcionalitats d'OpenGL necessaries per al projecte, un vèrtex shader "shader.vs" on no es fa pràcticament res, un fragment shader "shader.fs" on s'executa tot el codi de l'algorisme i representació de l'escena i una carpeta "dataStructures" on es troben les definicions de les estructures de dades que faciliten el procés. També es troben altres arxius secundaris que genera el Visual Studio a l'hora d'executar.

- Projecte OpenGL Linux: En aquest repositori es troba un "main.cpp" molt similar al del repositori de Windows, un vèrtex shader "shader.vs" igual que el del repositori de Windows, un fragment shader "shader.fs" igual que el del repositori de Windows, un Makefile que permet compilar i executar el projecte en Linux i la mateixa carpeta "dataStructures" que al repositori Windows.

- Projecte OpenGL i Qt: En aquest repositori es troben un "main.cpp" diferent del dels altres repositoris, un "shader.vs" similar al dels altres repositoris, un "shader.fs" molt similara al dels altres repositoris, la carpeta "dataStructures" i diverses classes que permeten el funcionament d'OpenGL i Qt alhora. Les classes són: "MainWindow", "Window", "GLWidget" i "Logo".

3.1.4 Qt

La configuració de Qt era un procediment que des del principi es considerava una ampliació molt interessant per al projecte, però no era segur que s'implementaria finalment. Per aquest motiu aquest procés s'ha realitzat durant l'última part del projecte. I com s'esperava, no ha sigut gens fàcil i ha calgut intentar múltiples formes d'afegir la llibreria Qt al projecte. El principal problema ha sigut que el projecte amb únicament OpenGL s'havia desenvolupat utilitzant les llibreries de 32bits, mentre que les implementacions de Qt actuals són de 64 bits.

El procediment que finalment va funcionar per afegir Qt al projecte és el següent:

1. Modificar les propietats de Microsoft Visual Studio 2017 per fer-lo compatible amb Microsoft Visual Studio 2015.

2. Afegir un plugin de Microsoft Visual Studio anomenat "Qt VS Tools" que permet crear projectes de Qt, crear arxius ".pro" i vincular els projectes a qualsevol versió de Qt instal·lada a l'ordinador.

3. Instal·lar una versió més antiga de Qt de 32 bits amb la que és compatible OpenGL de 32 bits. Aquesta versió de Qt no era compatible amb Microsoft Visual Studio 2017 (per això el primer pas era fer compatible Microsoft Visual studio 2017 amb la versió 2015).

4. Crear un projecte de Qt a Microsoft Visual Studio i crear les classes adequades a partir d'una referència[12] a un projecte OpenGL i Qt.

5. Adaptar les classes i l'escena per a llegir i fer funcionar els Shaders existents i fer-los funcionar correctament.

6. Modificar l'interfície per a modificar l'escena de la forma adequada.

3.2 Generació de l'escena base

Un dels primers passos un cop ja s'han instal·lat i configurat totes les eines és generar la primera escena a partir de la qual s'implementaran les millores visuals i realistes. Cal tenir en compte que en aquesta fase del projecte encara no s'havia inclòs Qt en el projecte.

Per a començar amb els primers passos d'OpenGL on es defineixen les bases del projecte, es criden als procediments principals i es llegeixen els shaders es va utilitzar una referència[7] on es donen exemples bàsics.

Un cop el procediment d'OpenGL funciona correctament, cal implementar una versió bàsica de l'algorisme per a mostrar escenes bàsiques a partir de les quals s'implementaran millores visuals i de rendiment. Per a veure una versió bàsica de l'escena vegeu la figura 1 a l'apartat "Abast", i per més informació de l'algorisme vegeu més endavant en el document a l'apartat "Anàlisi de les propietats de l'algorisme i l'entorn".

La primera millora visual, de la que es parlarà a continuació consisteix a augmentar el nombre de figures geomètriques que es poden representar a les escenes.

3.3 Figures Geomètriques

Una de les millores més importants a desenvolupar és la d'afegir possibilitats a l'usuari a l'hora d'omplir les seves escenes amb diverses figures geomètriques.

Més endavant s'explica el funcionament de l'algorisme i les Signed Distance Functions (SDF)[13], que són les funcions que s'utilitzen per a representar les diferents figures geomètriques. Aquestes funcions principalment indiquen a l'algorisme a on es troben els objectes de l'escena, quina forma tenen i quines dimensions tenen.

Els paràmetres que necessiten les figures geomètriques s'han modificat respecte a les funcions referenciades. Totes les figures tenen un material i alguns altres valors com les mides també s'han modificat en algunes funcions.

A continuació s'explica cadascuna de les figures geomètriques implementades i es mostren visualment. Cal tenir en compte que per a la generació de les imatges d'aquesta secció de figures geomètriques s'ha desactivat algunes de les implementacions visuals com: diverses fonts de llum, reflexions i càlcul d'ombres suaus.

3.3.1 Esfera

L'esfera és una de les figures principals i de les primeres a ser implementades. Per a cridar a la funció de l'esfera calen:

- Punt: variable a partir de la qual es calcularà la distància.
- Radi: radi de l'esfera a generar.
- Transformacions geomètriques: es passa una matriu o el resultat d'un càlcul de matrius representant totes les transformacions geomètriques que s'apliquen a l'objecte.
- Material: es passa un enter que representa un material.

La funció del càlcul de la distància a l'esfera és la següent:

- Es calcula el centre de l'esfera: p = (inverse(transfMatrix)* vec4(p, 1.0)).xyz;
- I la distància mínima entre el punt i l'esfera és: length(p)-s; (s és el radi).

Un cop passats tots els paràmetres es mostrarà l'esfera. Un exemple d'esfera es pot veure a continuació en la Figura 3.

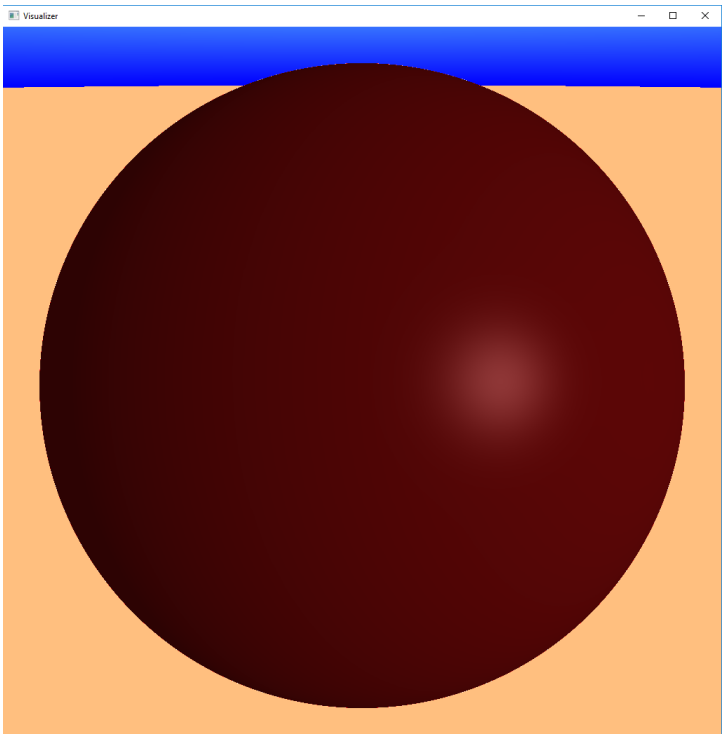


Figura 3: esfera centrada amb radi 1 i material 3 (rubí)

3.3.2 Cub

El cub també és una de les figures bàsiques que s'ha de trobar en el projecte. Per a cridar a la funció calen els següents paràmetres:

- Punt: variable a partir de la qual es calcularà la distància.
- Transformacions geomètriques: es passa una matriu o el resultat d'un càlcul de matrius representant totes les transformacions geomètriques que s'apliquen a l'objecte.
- Mides del cub en forma de vec3(mida component x/4, mida component y/4, mida component z/4)
- Material: es passa un enter que representa un material.

La funció del càlcul de la distància al cub és la següent:

- Es calcula el centre del cub: p = (inverse(transfMatrix)* vec4(p, 1.0)).xyz;
- Es tenen en compte les mesures del cub: vec3 d = abs(p) - mesures;

- I finalment es calcula la distància tenint en compte si el punt es troba dins o fora de l'objecte intentant evitar l'ús de condicions: min(max(d.x,max(d.y,d.z)),0.0) + length(max(d,0.0));

Un exemple de cub es pot veure a continuació en la Figura 4.

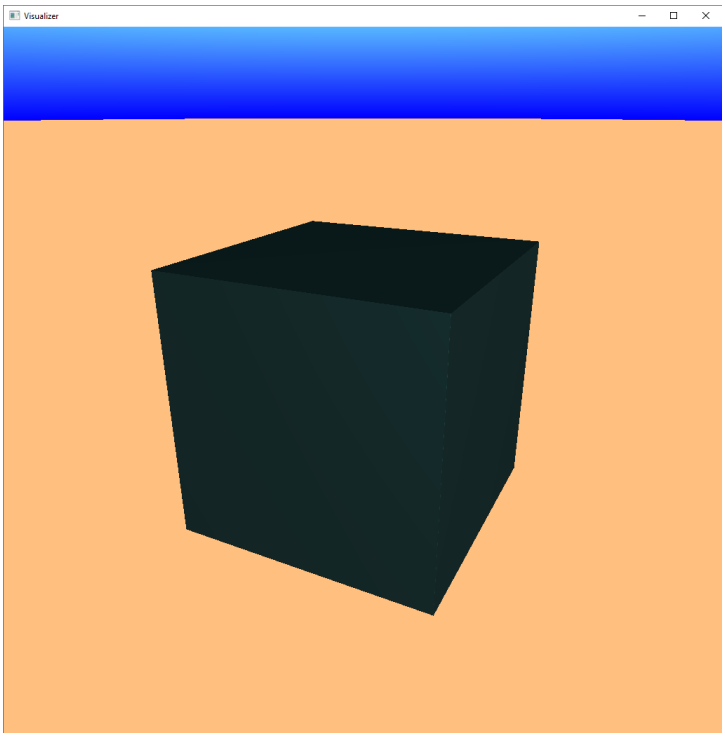


Figura 4: cub centrat amb mides reals: 4, 4, 4 i material 8 (goma de color cian)

3.3.3 Cilindres

Una altra figura que és interessant de poder representar és el cilindre. En aquest projecte hi ha 2 possibles cilindres a afegir a les escenes:

- Cilindre infinit: aquest tipus de cilindre té llargada infinita. Per a dibuixar-lo cal passar com a paràmetres: beginitemize
  - El punt a partir del qual es tornarà la distància.
  - Matriu de transformació.
  - Material de l'objecte.

Aquest cilindre també es pot convertir en finit, però per fer-ho cal utilitzar una funció d'intersecció amb algun altre objecte (aquesta funció es pot veure una mica més endavant en el document). Es pot veure un exemple d'aquest cilindre a la Figura 5.

- Cilindre finit: aquest cilindre té més possibles manipulacions que l'anterior, com el radi i la llargada. Els paràmetres són:
  - El punt a partir del qual es tornarà la distància.
  - vec2 amb el radi del cilindre i l'alçada d'aquest.
  - Matriu de transformació.
  - Material de l'objecte.

  A la Figura 6 hi ha un exemple d'aquest cilindre.

La funció del càlcul de la distància al cilindre infinit és la següent:

- Es calcula el centre del cilindre: p = (inverse(transfMatrix)* vec4(p, 1.0)).xyz;
- I es calcula la distància sense tenir en compte el valor de la y (ja que el cilindre té llargada infinita). I se li resta el radi del cilindre: length(p.xz)-c.z;

I la funció del càlcul de la distància al cilindre finit és la següent:

- Es calcula el centre del cilindre: p = (inverse(transfMatrix)* vec4(p, 1.0)).xyz;
- Es guarden les mesures del cilindre: vec2 d = abs(vec2(length(p.xz),p.y)) - h;
- I finalment, evitant condicionals, es calcula la distància a l'objecte: min(max(d.x,d.y),0.0) + length(max(d,0.0));

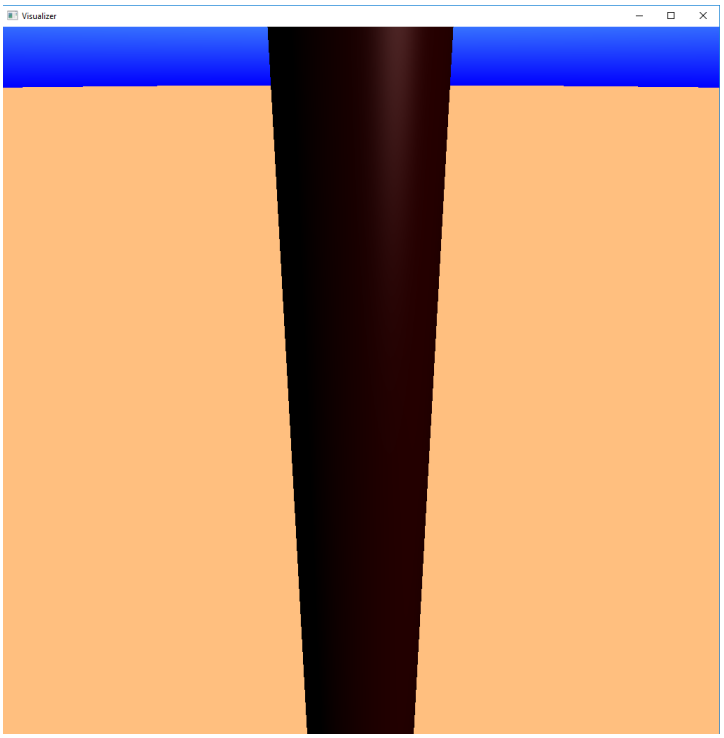


Figura 5: cilindre infinit amb material 4 (plàstic vermell)

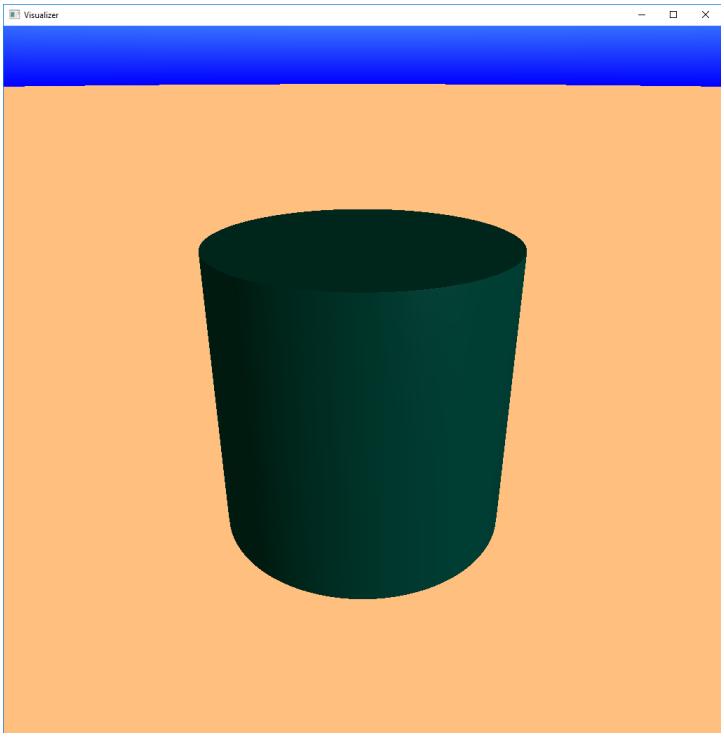


Figura 6: cilindre finit amb radi 1, llargada 1 i material 5 (plàstic cian)

3.3.4 Tor

Una altra de les figures implementades és el tor. Els paràmetres necessaris per a dibuixar-lo són:

- El punt a partir del qual es tornarà la distància.
- vec2 amb el radi del forat central i el radi del tor.
- Matriu de transformació.
- Material de l'objecte.

La funció del càlcul de la distància al tor és la següent:

- Es calcula el centre del tor: p = (inverse(transfMatrix)* vec4(p, 1.0)).xyz;
- Es tenen en compte les mesures del tor: vec2 q = vec2(length(p.xz)-t.x,p.y);
- I finalment es calcula la distància: length(q)-t.y;

A la figura 7 es pot trobar l'exemple d'un tor dibuixat pel programa.

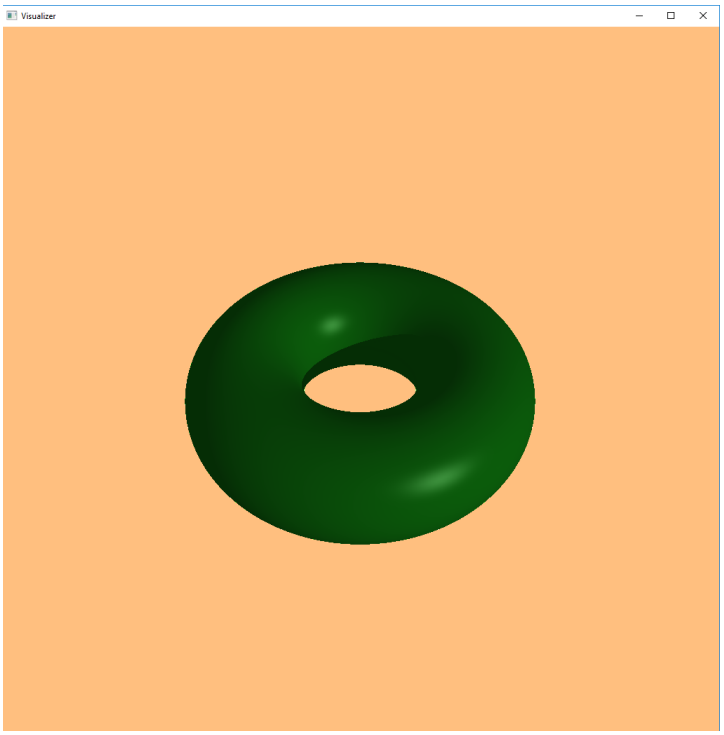


Figura 7: tor amb radi del forat 1, radi del tor 0.5 i material 1 (maragda)

3.3.5 Con infinit

En el projecte també es poden representar cons. El con implementat és infinit per això, i per això cal realitzar la intersecció amb alguna altra figura si es vol una mica concreta del con. Els paràmetres per a dibuixar aquesta figura són:

- El punt a partir del qual es tornarà la distància.
- Matriu de transformació.
- Material de l'objecte.

La funció del càlcul de la distància al con és la següent:

- Es calcula el centre del con: p = (inverse(transfMatrix)* vec4(p, 1.0)).xyz;
- Es guarda la distància de les components x i y del punt: float q = length(p.xy);
- I finalment es calcula el producte escalar entre les mesures(c), el càlcul anterior(q) i la component z del punt: dot(c,vec2(q,p.z));

A la Figura 8 es pot trobar un con infinit dibuixat pel programa.

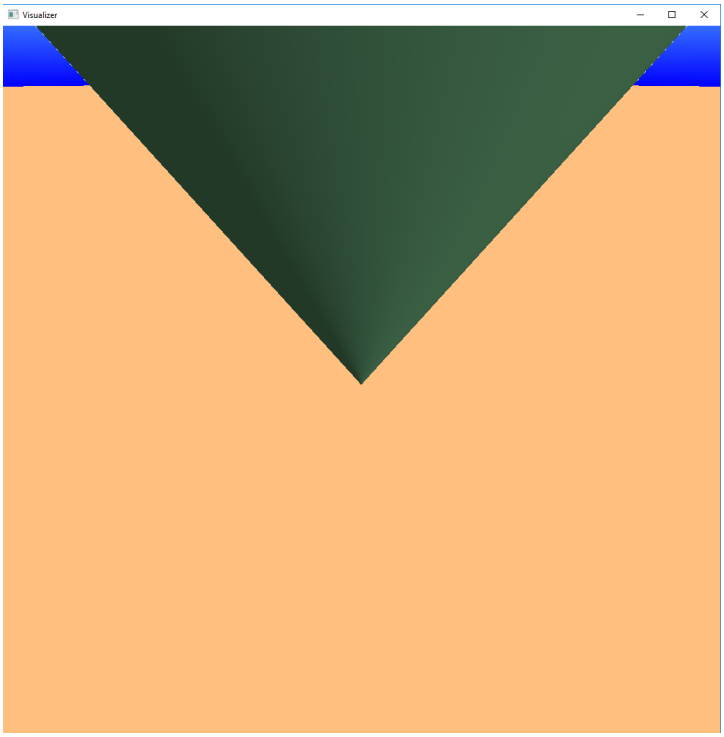


Figura 8: con infinit amb material 2 (jade)

3.3.6 Unió

Un cop totes les figures interessants ja es poden representar a les escenes, es va considerar molt interessant disposar també d'operacions entre figures. La unió és una d'aquestes operacions.

Aquesta operació principalment consisteix a dibuixar més d'un objecte alhora completament. En el punt de vista algorísmic bàsicament consisteix a retornar la distància mínima als 2 objectes que pertanyen a la unió. També és la tècnica que s'utilitza per a dibuixar escenes amb molts objectes. Un exemple d'una escena amb una esfera, cub1, cub2 i cilindre seria: unió (esfera, unió (cub1, unió (cub2, cilindre))).

I com s'ha vist en el petit exemple just a sobre, aquesta funció rep com a paràmetre el retorn de 2 funcions de qualsevol figura.

A la Figura 9 es pot veure un exemple d'unió entre un cub i una esfera.

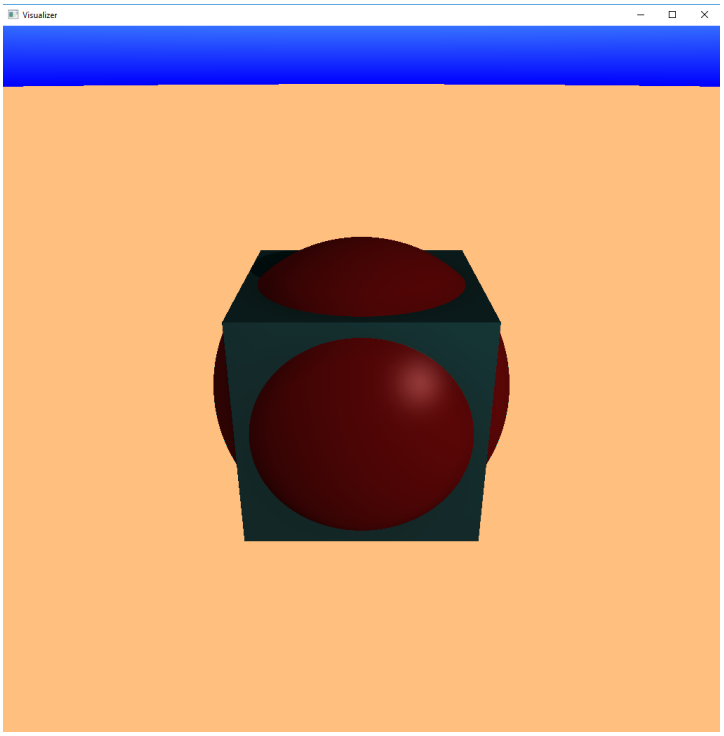


Figura 9: unió entre esfera de material 3 (rubí) i cub de material 8 (goma de color cian)

3.3.7 Intersecció

Una altra de les operacions entre figures que es poden utilitzar és la intersecció. Que com es pot intuir, només es dibuixa la part entre 2 objectes que pertany als 2 objectes alhora. Algorísmicament es retorna la distància màxima entre els 2 objectes que formen part de la funció.

La usabilitat és la mateixa que la unió: intersecció (objecte 1, objecte 2).

A la Figura 10 es pot veure un exemple de la intersecció entre una esfera i un cub.

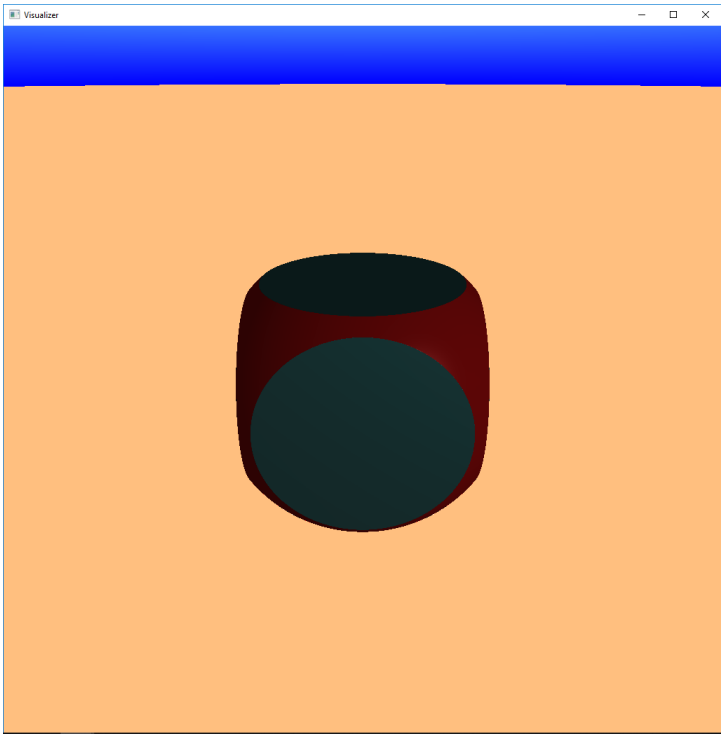


Figura 10: intersecció esfera de material 3 (rubí) i cub de material 8 (goma de color cian)

3.3.8 Diferència

I per acabar les operacions entre figures també s'ha implementat la diferència. El format és el mateix que en la unió i la intersecció però en aquest cas és important l'ordre dels components. Algorísmicament la funció retorna el màxim entre objecte 1 i l'objecte 2 negatiu.

No és el mateix calcular la diferència (objecte 1, objecte 2) que diferència (objecte 2, objecte 1). En les Figures 11 i 12 es pot veure un exemple de diferència esfera - cub i cub - esfera.

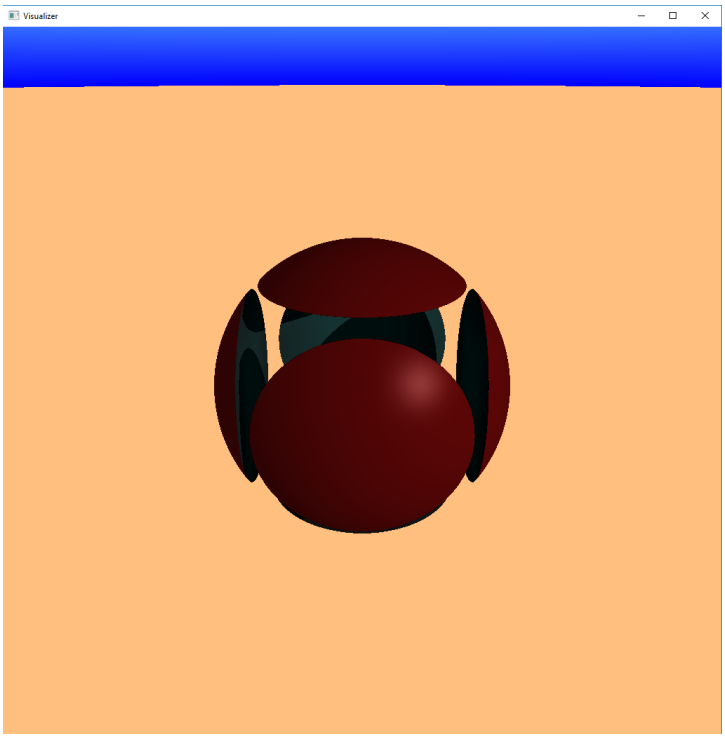


Figura 11: diferència entre esfera de material 3 (rubí) - cub de material 8 (goma de color cian)

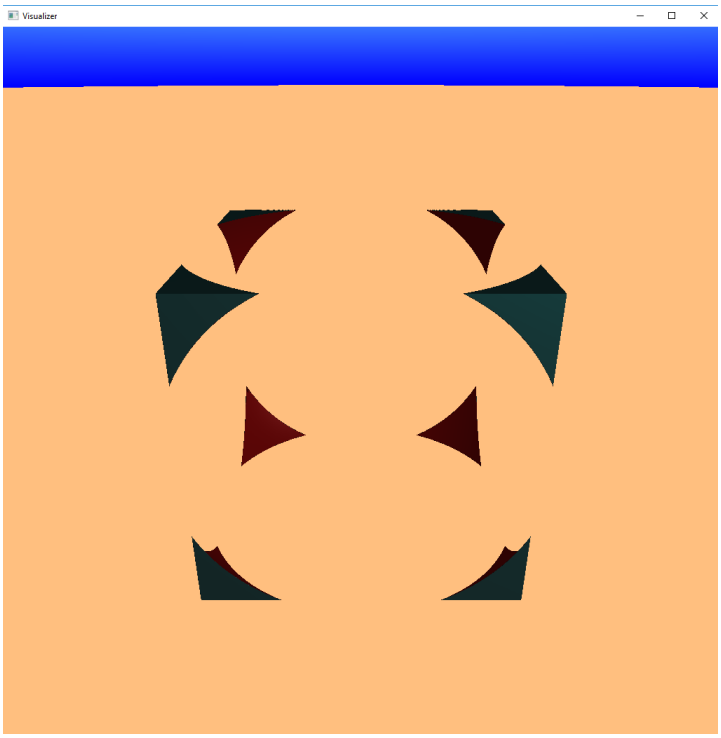


Figura 12: diferència entre cub de material 8 (goma de color cian) - esfera de material 3 (rubí)

3.4 Transformacions geomètriques

Una altra de les implementacions que permeten manipular les escenes són les transformacions geomètriques. En la secció anterior s'ha estat parlant sobre l'ús d'aquestes transformacions, però he decidit parlar de les transformacions després de parlar de les figures perquè realment va ser una implementació posterior a les figures.

Es va considerar una ampliació molt interessant la de poder aplicar translacions i rotacions a qualsevol objecte independentment i en qualsevol ordre. Per aquest mateix motiu, es van implementar.

Per a fer funcionar aquesta característica s'han implementat les següents funcions:

- translation(x, y, z): aquesta funció retorna la matriu 4x4 que representa la translació en x, y i z que indiquen els paràmetres.
- rotation(eix, angle): funció que retorna la matriu 4x4 que representa la rotació en l'eix indicat de l'angle indicat en els paràmetres.

Amb aquestes dues funcions implementades es poden representar transformacions prou complexes. Per exemple, si es vol dibuixar un cub amb una rotació i després aplicar-li una translació (l'ordre és important), s'hauria de fer de la següent forma (pseudo codi):

cub(punt, translation(x, y, z)*rotation(eix, angle), mides, material).

En les figures 13 i 14 es pot veure un cub sense transformacions i un cub amb transformacions.

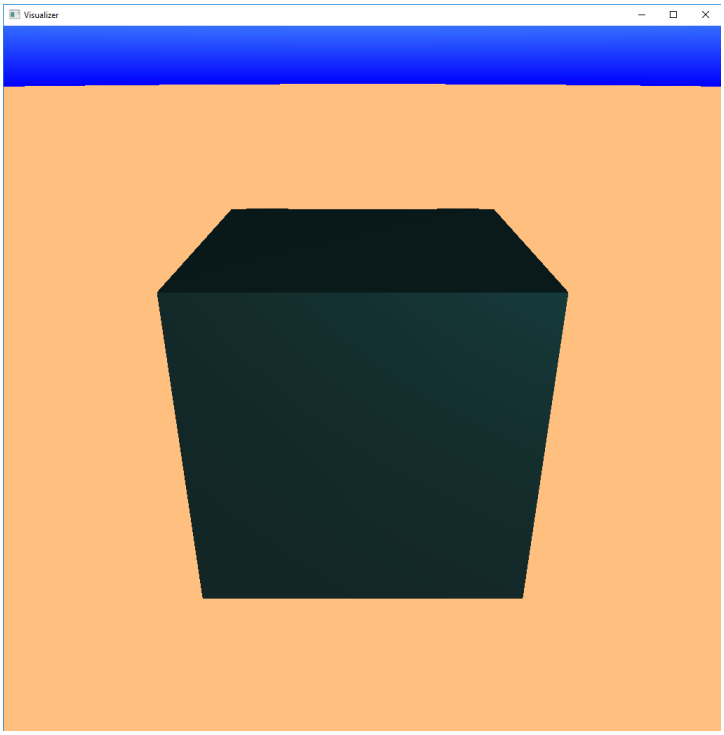


Figura 13: cub de material 8 (goma de color cian) sense transformacions.

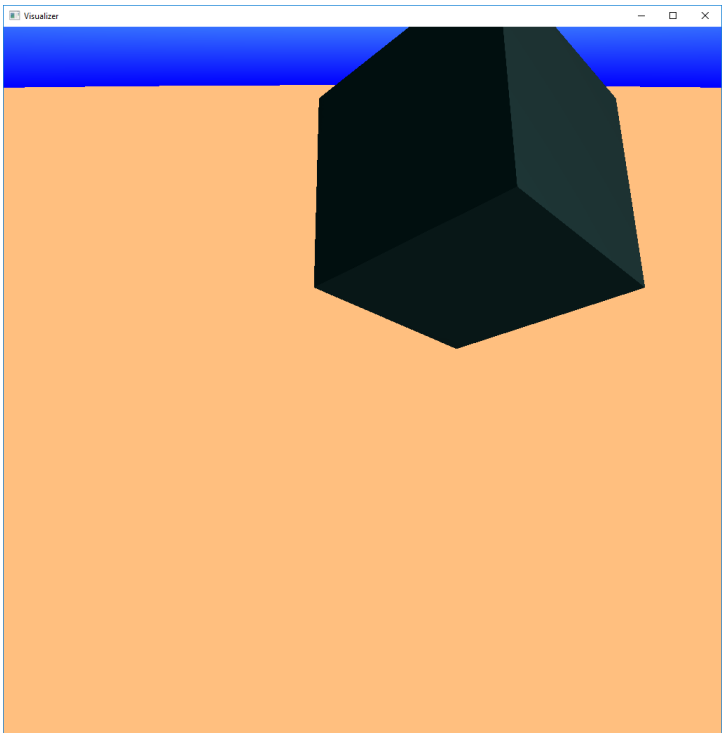


Figura 14: cub de material 8 (goma de color cian) amb les següents transformacions aplicades en l'ordre següent: rotació en l'eix Y de -45º, rotació en l'eix X de 45º i translació de (1, 1, -2) unitats.

3.5 Materials

També es va considerar molt interessant tenir la possibilitat que cadascuna de les figures tingui un material que determini diverses propietats visuals: com el color i la reflexió d'aquest. Els materials implementats són els següents:

1. Maragda.
2. Jade.
3. Rubí.
4. Plàstic vermell.
5. Plàstic cian.
6. Plàstic verd.

7. Goma vermella.
8. Goma cian.
9. Goma verda.
10. Ciment.
11. Goma groga.
12. Mirall.

Cadascun d'aquests materials té la seva pròpia component ambient, difosa, especular i coeficient de reflexió que determinen el color que tindrà l'objecte. També tenen el seu propi coeficient de reflexió.

A continuació es poden veure els valors de cadascuna de les 4 components per a cada material. Cada component té 3 valors que corresponen a la component del color RGB:

3.5.1 Component ambient

1. Maragda: vec3(0.0215, 0.1745, 0.0215).
2. Jade: vec3(0.135, 0.2225, 0.1575).
3. Rubí: vec3(0.1745, 0.01175, 0.01175).
4. Plàstic vermell: vec3(0.0, 0.0, 0.0).
5. Plàstic cian: vec3(0, 0.1, 0.06).
6. Plàstic verd: vec3(0, 0, 0).
7. Goma vermella: vec3(0.05, 0, 0).
8. Goma cian: vec3(0, 0.05, 0.05).
9. Goma verda: vec3(0, 0.05, 0).
10. Ciment: vec3(0.25, 0.20725, 0.20725).
11. Goma groga: vec3(0.4, 0.32, 0.2).
12. Mirall: indiferent.

3.5.2 Component difosa

1. Maragda: vec3(0.07568, 0.61424, 0.07568).
2. Jade: vec3(0.54, 0.89, 0.63).
3. Rubí: vec3(0.61424, 0.04136, 0.04136).
4. Plàstic vermell: vec3(0.5, 0, 0).
5. Plàstic cian: vec3(0, 0.51, 0.51).
6. Plàstic verd: vec3(0.1, 0.35, 0.1).
7. Goma vermella: vec3(0.5, 0.4, 0.4).
8. Goma cian: vec3(0.4, 0.5, 0.5).
9. Goma verda: vec3(0.4, 0.5, 0.4).
10. Ciment: vec3(1, 0.829, 0.829).
11. Goma groga: vec3(0.7, 0.62, 0.4).
12. Mirall: indiferent.

3.5.3 Component especular

Aquesta component té un quart valor que correspon al "shininess" del material, que fa referència a la facilitat amb la qual es veu la taca especular i la mida d'aquesta.

1. Maragda: vec4(0.633, 0.727811, 0.633, 0.6).
2. Jade: vec4(0.316, 0.316, 0.316, 0.1).
3. Rubí: vec4(0.728, 0.627, 0.627, 0.6).
4. Plàstic vermell: vec4(0.7, 0.6, 0.6, 0.25).
5. Plàstic cian: vec4(0.5, 0.5, 0.5, 0.25).
6. Plàstic verd: vec4(0.45, 0.55, 0.45, 0.25).

7. Goma vermella: vec4(0.7, 0.04, 0.04, 0.078).
8. Goma cian: vec4(0.04, 0.7, 0.7, 0.078).
9. Goma verda: vec4(0.04, 0.7, 0.04, 0.078).
10. Ciment: vec4(0.29, 0.29, 0.29, 0.088).
11. Goma groga: vec4(0.8, 0.8, 0.5, 0.078).
12. Mirall: indiferent.

3.5.4 Component reflexió

1. Maragda: vec3(0.06,0.06, 0.06).
2. Jade: vec3(0.06, 0.06, 0.06).
3. Rubí: vec3(0.06, 0.06, 0.06).
4. Plàstic vermell: vec3(0.08, 0.08, 0.08).
5. Plàstic cian: vec3(0.08, 0.08, 0.08).
6. Plàstic verd: vec3(0.08, 0.08, 0.08).
7. Goma vermella: vec3(0.05, 0.05, 0.05).
8. Goma cian: vec3(0.05, 0.05, 0.05).
9. Goma verda: vec3(0.01, 0.01, 0.01).
10. Ciment: vec3(0.001, 0.001, 0.001).
11. Goma groga: vec3(0.02, 0.02, 0.02).
12. Mirall: vec3(1, 1, 1).

En l'apartat anterior de figures geomètriques es poden veure diversos exemples de materials diferents, i en apartats posteriors on les escenes tenen més càlculs també es poden veure exemples on cada material es comporta de forma diferent.

3.6 Llums i ombres

Quan es vol implementar una aplicació gràfica realista, és pràcticament obligatori que en alguna part d'aquest procés s'ha desenvolupat la il·luminació, tractament de llums i ombres. És una implementació de les que més sensació de realisme aporta.

En aquest projecte s'ha implementat el càlcul habitual de la llum amb les components ambient, difosa i especular, un càlcul d'ombres, més endavant es parlarà de les ombres suaus i finalment a les escenes es permet tenir fins a 3 fonts de llum amb intensitat variable. La raó per la qual es permeten fins a 3 fonts de llum és per poder utilitzar la tècnica de generació d'escenes d'il·luminació de 3 punts[15].

Per a calcular el color cada material té uns valors de RGB ambient, difós i especular que segons la posició del píxel tractat, la càmera i la llum s'utilitzarà un d'aquests o tots. Per exemple: en el cas que al punt que s'està tractant no arribi cap font de llum, només s'utilitzarà la component ambient. En canvi, si hi arriba la llum i està a la posició adequada, al punt se sumarà les 3 components i aquell punt en concret tindrà una saturació de color notable.

A continuació es descriurà les propietats del càlcul de llum:

3.6.1 Component ambient

En els gràfics la component ambient és un color que arriba a tot arreu, fins i tot als llocs on no arriba cap tipus de llum. Com que és una il·luminació que està sempre present, el valor d'aquesta component ha de ser baix. Per a calcular la component ambient de l'escena, cada material té un valor RGB bastant baix per a aquesta component que s'aplica sempre a tots els punts.

3.6.2 Component difosa

La component difosa es calcula quan la font de llum arriba directament. Igual que en el cas de l'ambient, cada material té una component difosa diferent. Com que aquesta component s'aplica quan arriba la llum directament, els valors del color són majors que els de la component ambient.

3.6.3 Component especular

Per acabar les components, l'especular és la "taca" de color més saturat que es veu en els objectes quan hi arriba la llum. En aquest el valor que tenen els materials per a aquesta component és considerablement més gran.

Per a calcular si el punt en concret està influït per la "taca" especular, hi ha un procés de càlculs indicat. A la referència[3] es pot veure una petita guia per a realitzar els càlculs.

3.6.4 Raig de llum

En aquesta secció s'ha estat parlant sobre si a un punt arriba o no la llum, però l'algorisme implementat només fa arribar el raig fins a l'objecte i un cop acabat aquest procés no se sap si la llum arriba o no. El que s'ha implementat és un procés posterior al de l'algorisme que traça un raig des del punt cap a cadascuna de les fonts de llum. I un cop se sap quantes fonts de llum arriben al punt, se suma la component difosa els cops que faci falta i la component especular en cas que sigui adient.

3.6.5 Ombres

L'efecte de les ombres afecta molt de realisme i és molt interessant. Per a aconseguir aquest efecte basta que els càlculs esmentats en els punts anteriors funcionin correctament i ja es tindrà les ombres implementades.

3.6.6 Ombres suaus

Un altre efecte que millora la qualitat de les llums és el de les ombres suaus, tot i que aquest càlcul no és tan senzill com el de les ombres normals. Aquest efecte simula que les fonts de llum tenen una mida i que no els fotons de tota la superfície de la llum arriben als mateixos llocs.

Per a simular aquest efecte i calcular si un punt ha de formar part d'una ombra suau, durant el procés d'anar des del punt a dibuixar fins a la llum es manté constància si s'ha passat molt a prop d'algun altre objecte. Si s'ha passat a prop d'algun objecte, aquest punt tindrà la il·luminació una mica reduïda i simularà una ombra suau.

Els coeficients de distància mínima i llindars d'aquest procés no són gens fàcils de determinar i depenen molt de les propietats de l'escena.

A les figures 15 i 16 es pot veure una escena amb els càlculs de llum sense ombres suaus i la mateixa escena amb les ombres suaus.

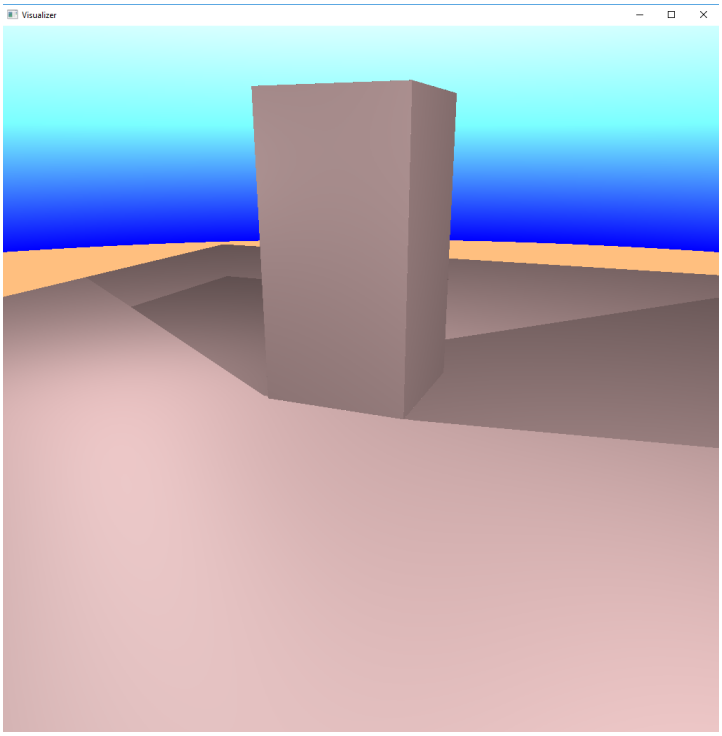


Figura 15: Escena amb 3 fonts de llum sense utilitzar el càlcul d'ombres suaus.

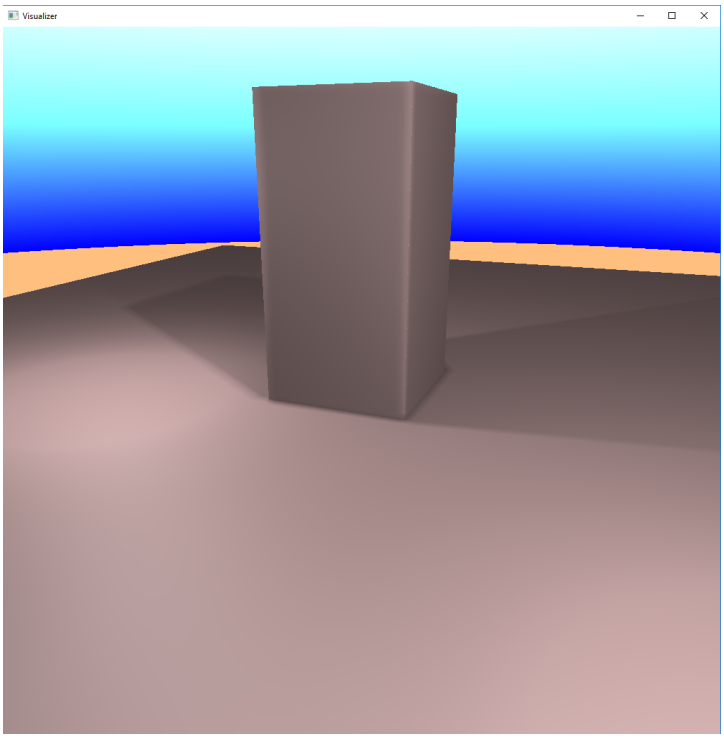


Figura 16: La mateixa escena que a la Figura 15 però amb càlcul d'ombres suaus activat.

3.7 Escenes més complexes

Un cop ja es té una base de realisme i de figures a representar, és interessant també tenir algunes escenes on es demostri les possibilitats del programa. Per fer això es va decidir dissenyar i implementar les següents 2 escenes:

- Per una banda és interessant tenir una escena on es poden veure totes (o gran part) de les funcionalitats i possibles figures a representar. Per això es va fer l'escena que es pot veure a la Figura 17.
- I per altra banda també és molt interessant tenir una escena on se simula un entorn més complex. A la Figura 18 es pot veure un exemple de l'escena.

Cal tenir en compte que les escenes estan representades només amb les implementacions visuals que s'han explicat fins ara.

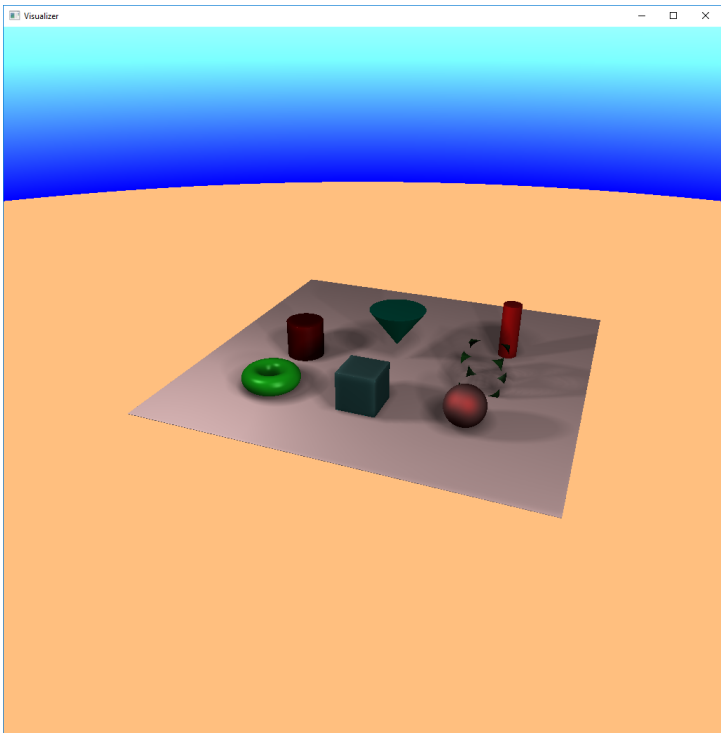


Figura 17: escena amb gran part de les figures possibles a representar.

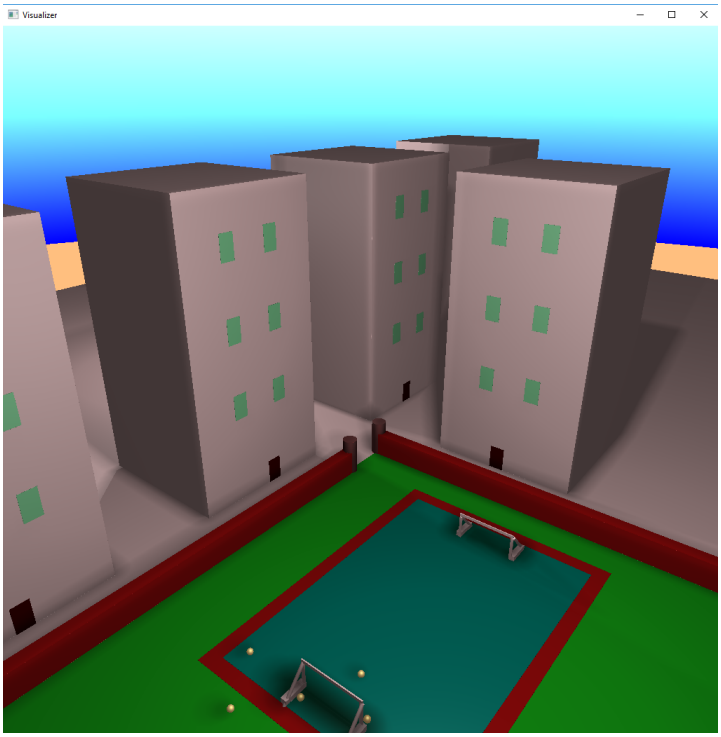


Figura 18: escena complexa.

3.8 Ambient occlusion

L'ambient occlusion és una tècnica que simula el grau d'exposició d'un punt a la il·luminació ambiental i reflexions dels objectes propers. Aquesta tècnica proporciona sensació de realisme i, per tant, és interessant d'implementar.

Una forma de calcular una aproximació d'aquest efecte és la següent: un cop s'arriba al punt de l'objecte, es projecta un raig en direcció de la normal i aquest fa una crida a la funció de distàncies (SDF). A partir del resultat de la funció es pot saber una aproximació de si l'objecte està en una zona oberta o no.

L'aproximació esmentada s'obté de la següent manera:

obscurancia = (epsilonOcclusion-objectesEscena(pAux).x)/epsilonOcclusion;

on "epsilonOcclusion" és el valor llindar a partir del qual es comença a representar l'efecte visual, i "objectesEscena" és la funció que retorna la distància mínima. Un cop es té el resultat, es multiplica el color per (1-obscurancia) de tal manera que el color es veu més fosc si es compleix la condició.

A les Figures 19 i 20 es poden veure exemples d'aquest efecte.

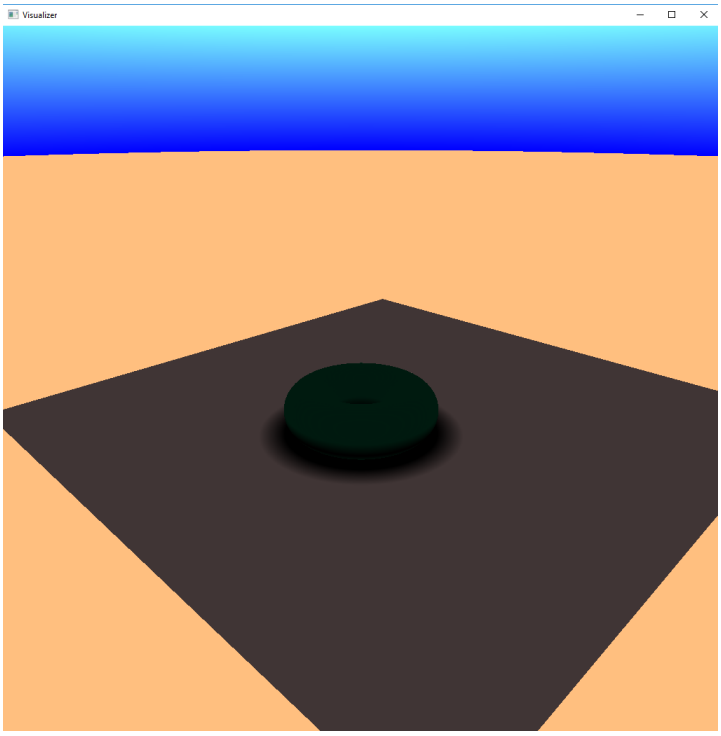


Figura 19: escena amb un tor sobre un pla i les fonts de llum amb intensitat 0.

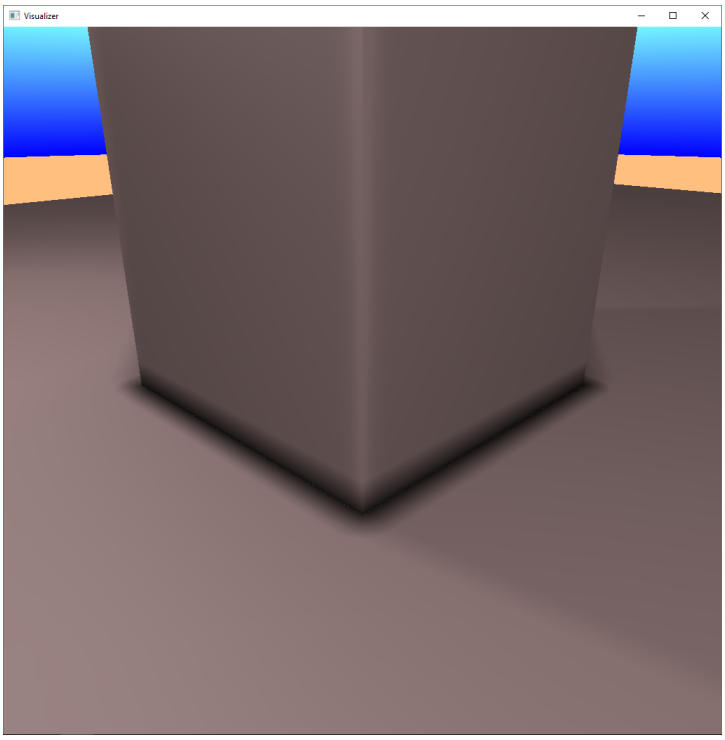


Figura 20: escena amb les llums activades on es pot veure una intersecció d'objectes i el resultat del càlcul d'ambient occlusion.

3.9 Moviment

Una forma de generar efecte de sensació de temps real és la implementació d'animacions dels objectes. Una forma de fer-ho és implementant un rellotge, fer que la posició d'una figura geomètrica vagi en funció del temps del rellotge i actualitzar els shaders periòdicament.

3.10 Reflexions

Una de les millores visuals més interessants és la implementació de les reflexions. Malgrat que aquesta implementació no és trivial, millora molt la qualitat dels resultats. Per tant, es va decidir dur-la a terme.

Una altra cosa a tenir en compte en aquest efecte és que els rajos no es poden comunicar entre ells. Això és important perquè per a aquest procés cal calcular el color de l'objecte que es reflecteix. I com que no hi ha comunicació, aquest color s'ha de re calcular. El procés de reflexió s'inicia un cop s'ha calculat el color del punt i s'ha realitzat el càlcul de llum, i el procés consisteix dels següents passos:

1. Càlcul del raig de reflexió a partir del qual s'agafarà el color final i es projectarà al punt en un percentatge. Aquest raig es pot calcular utilitzant la funció "reflect(vector incident, normal en el punt)". I bàsicament el que calcula la funció és: I - 2.0 * dot(N, I) * N;

2. Procés iteratiu de projecció del raig utilitzant un algorisme similar al de Ray Marching.

3. Càlcul complet del color del punt al qual arriba el raig.

4. Càlcul de l'efecte Fresnel (Procés que s'explica al següent apartat).

3.10.1 Efecte Fresnel

Hi ha un altre efecte realista que es pot aplicar a les reflexions i que és raonable d'implementar. Aquest és l'efecte Fresnel, i consisteix a variar el coeficient de reflexió de l'objecte depenent de l'angle de visió amb l'observador. Com més petit és l'angle de visió, més coeficient de reflexió ha de tenir el material. A la referència[5] (a partir de la pàgina 59) es pot veure exemples de materials i les propietats d'aquest efecte.

La fórmula utilitzada per a calcular el comportament del material s'anomena aproximació de Schlick. I és la següent:

1. Inicialment es calcula l'angle des d'on s'observa l'objecte: float angleFresnel = dot(normalize(vObs-puntcolisio), normal);

2. A continuació a partir d'aquest angle es calcula el coeficient de reflexió que tindrà l'objecte tenint en compte l'angle i l'índex de reflexió d'aquest material: vec3 coefReflexio = indexReflexMaterial + (1 - indexReflexMaterial)*pow((1 - angleFresnel), 5);

3. I finalment es calcula el color utilitzant els valors calculats: color = (1-coefReflexio)*color + coefReflexio*colorRef;

A la Figura 21 es pot veure una escena amb les reflexions sense aplicar l'efecte Fresnel, a la Figura 22 es pot veure la mateixa escena on en comptes de colors es fa servir el resultat del càlcul de Fresnel i a la Figura 23 es veu l'escena utilitzant tanta reflexió com Fresnel.

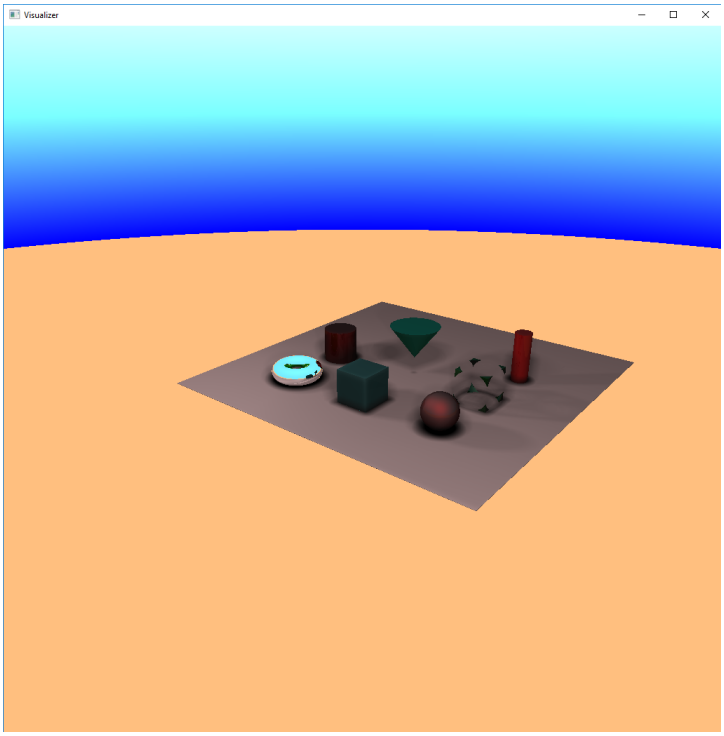


Figura 21: escena utilitzant només el càlcul bàsic de reflexions.

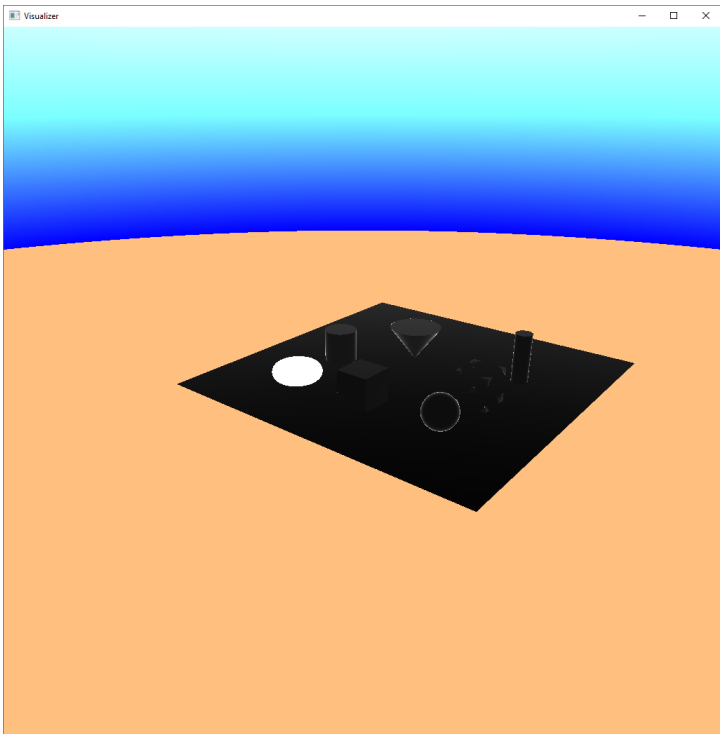


Figura 22: escena utilitzant el resultat del càlcul de l'efecte Fresnel. Les zones més blanques seran més reflectides.

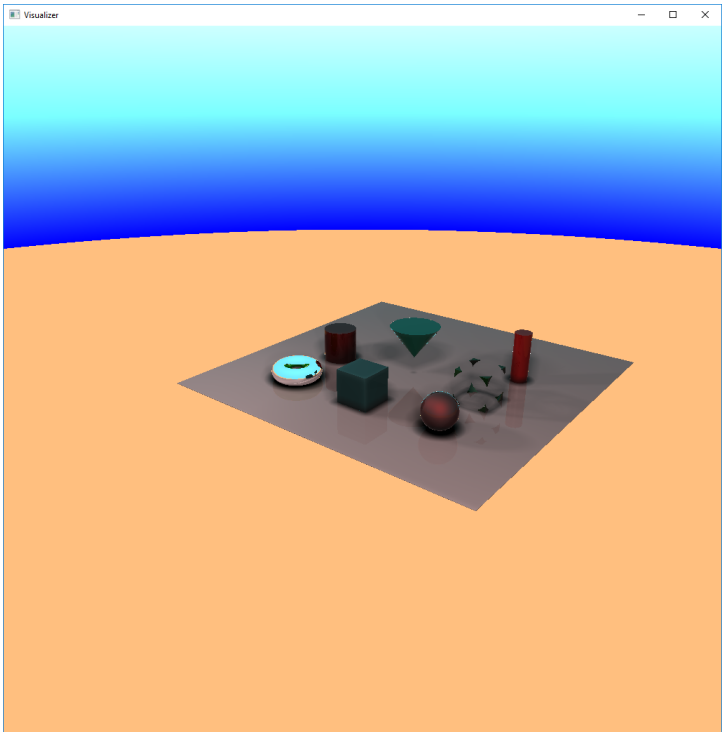


Figura 23: escena utilitzant ambdós càlculs. Es pot veure que l'angle de visió té efecte en la reflexió encara que el material sigui poc reflexiu.

3.11 Interfície

Un cop implementades totes les aplicacions esmentades, el projecte ja ofereix uns resultats prou realistes i es pot començar a treballar en el desenvolupament d'una interfície utilitzant Qt.

Com s'ha esmentat anteriorment, el procés de compatibilitzar Qt amb OpenGL no sempre és trivial i en aquest projecte en concret va ser un procés considerablement costós.

Un cop compatibles per això, hi ha diverses funcionalitats de les esmentades anteriorment que es poden controlar fàcilment mitjançant la interfície:

- Intensitat de cadascuna de les 3 fonts de llum de l'escena: hi ha 3 esliders que permeten modificar la intensitat de cadascuna de les 3 llums. Per veure el resultat del canvi de valors de les llums cal apretar el botó inferior "update".

- Reflexions: hi ha un "checkbox" que permet activar i desactivar el càlcul de les reflexions a l'escena. L'escena s'actualitza automàticament en canviar el valor d'aquest.
- Ombres suaus: "checkbox" que permet activar i desactivar el càlcul de les ombres suaus a l'escena. L'escena s'actualitza automàticament en canviar el valor d'aquest.
- Ambient occlusion: "checkbox" que permet activar i desactivar el càlcul de l'ambient occlusion de l'escena. L'escena s'actualitza automàticament en canviar el valor d'aquest.
- Actualització temps real: "checkbox" que activat fa que l'escena s'actualitzi 20 cops per segon. Aquest està per defecte desactivat perquè el seu objectiu principal és per a les escenes que contenen animacions. I si l'escena és massa completa i/o la màquina que l'executa no és prou potent, no ofereix un resultat prou fluït.

A la Figura 24 es pot veure una escena amb la interfície descrita.

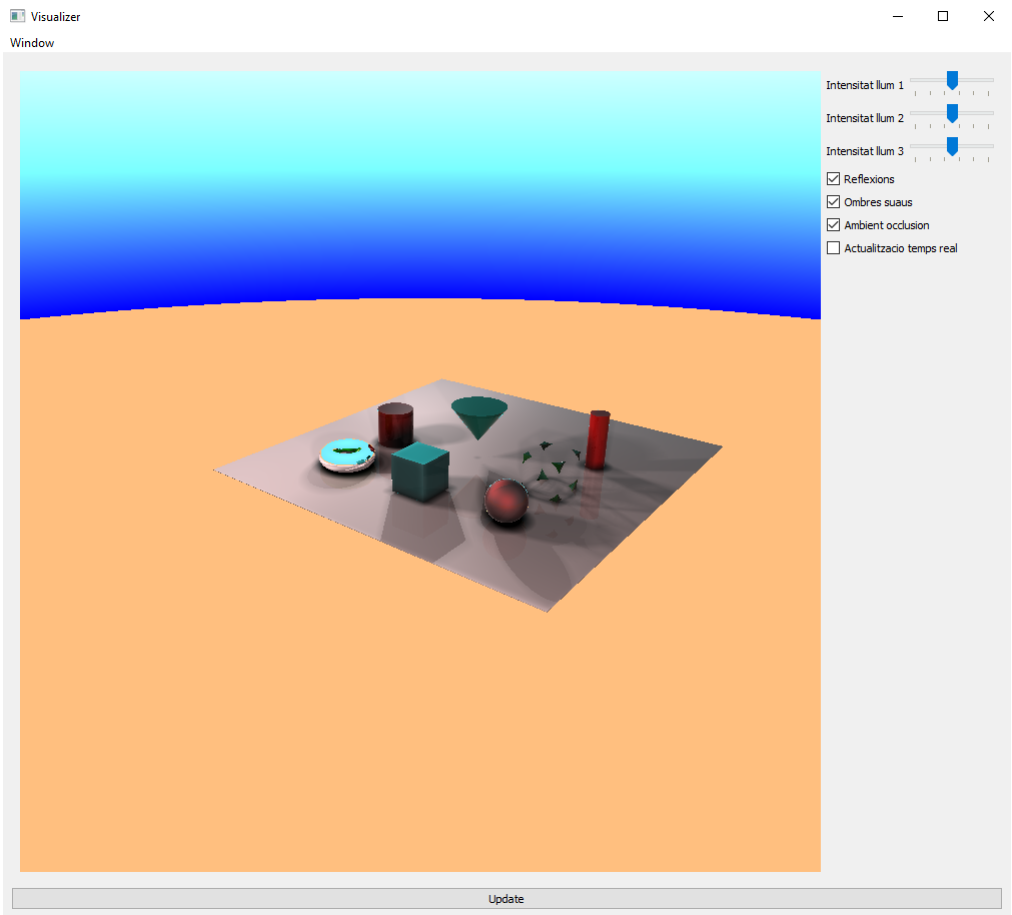


Figura 24: exemple d'escena amb la seva interfície.

4 Resultats

Un cop hi ha implementades totes les millores visuals esmentades en els apartats anteriors, ja es té un producte que compleix els objectius del projecte. A continuació hi ha un seguit de Figures (25-) mostrant el resultat final utilitzant una escena prou complexa i el funcionament de la interfície en aquesta:

- A la Figura 25 hi ha el resultat amb les 3 fonts de llum al màxim de potència. Es pot veure la saturació de colors perquè arriba massa llum als objectes.
- A la Figura 26 es pot veure el resultat amb les fonts de llum al mínim, on tots els objectes només tenen el color ambient i l'ambient occlusion.
- A la Figura 27 es pot veure el resultat agafant la configuració de la figura 26 i desactivant l'ambient occlusion.
- A la Figura 28 es pot veure el resultat amb un valor de les fonts de llum adequat per a simular la il·luminació de 3 punts.

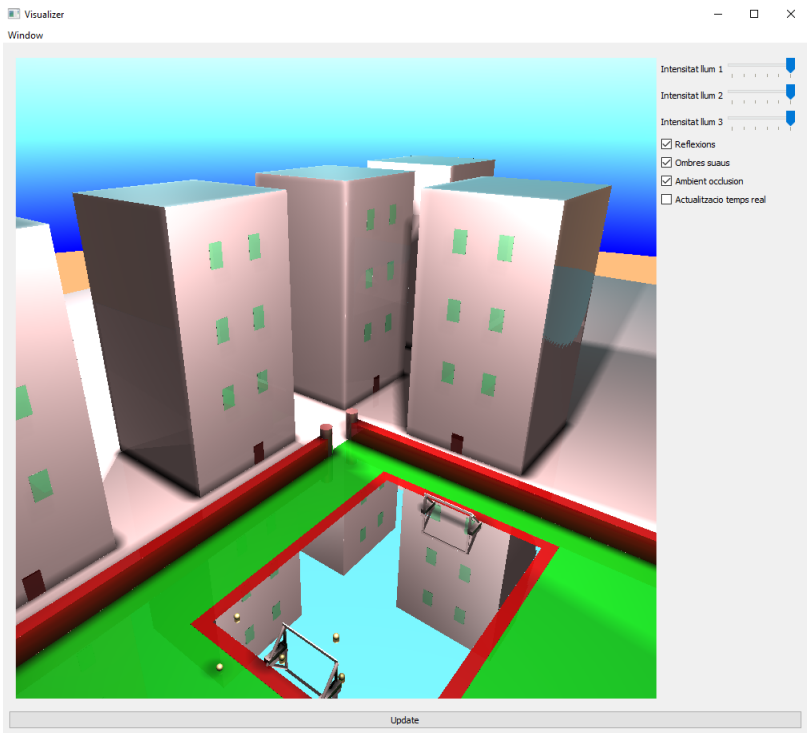


Figura 25: imatge amb tanta il·luminació que es veuen els colors molt saturats.

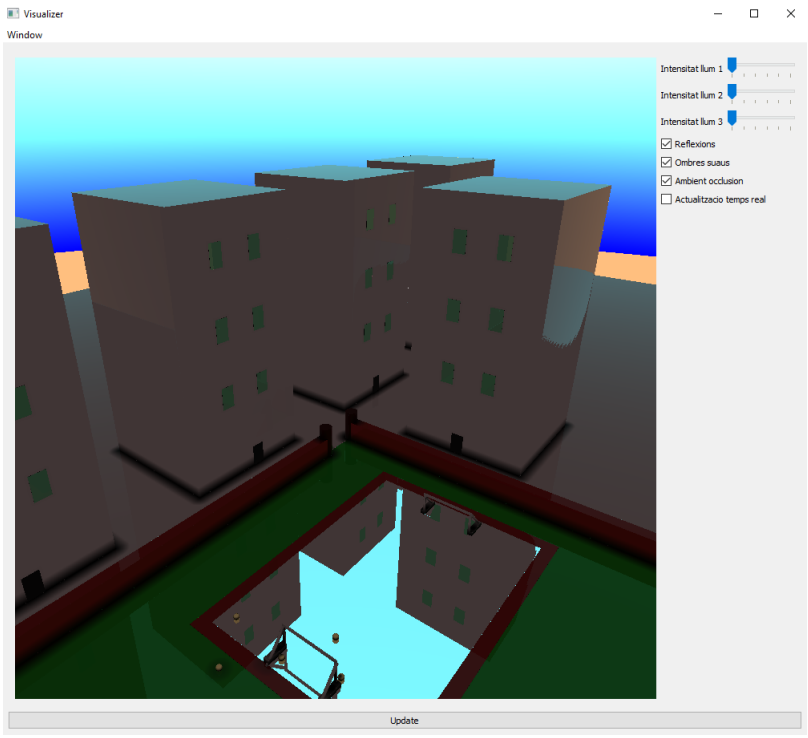


Figura 26: imatge sense il·luminació.

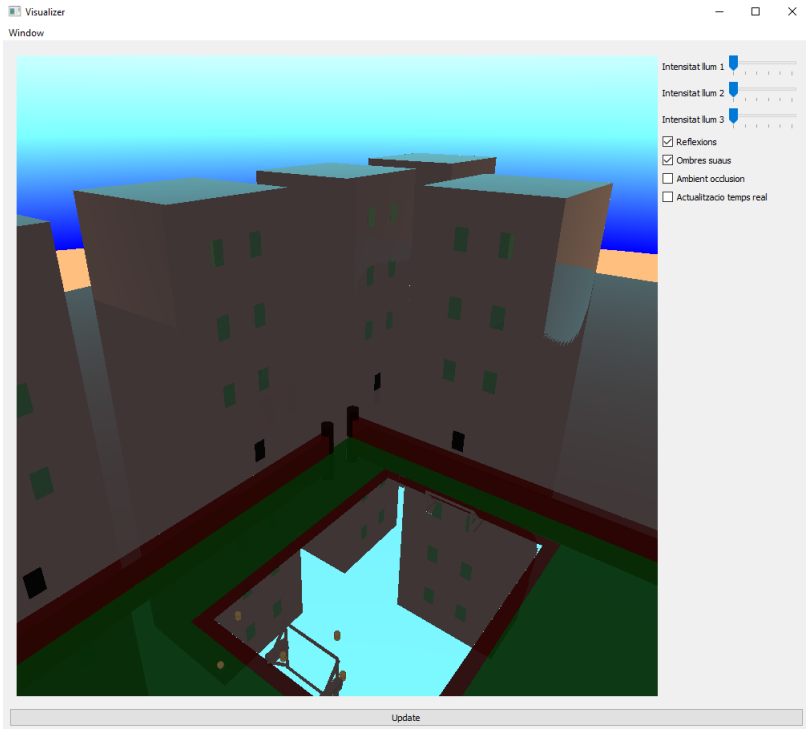


Figura 27: imatge sense il·luminació i ambient occlusion desactivat.

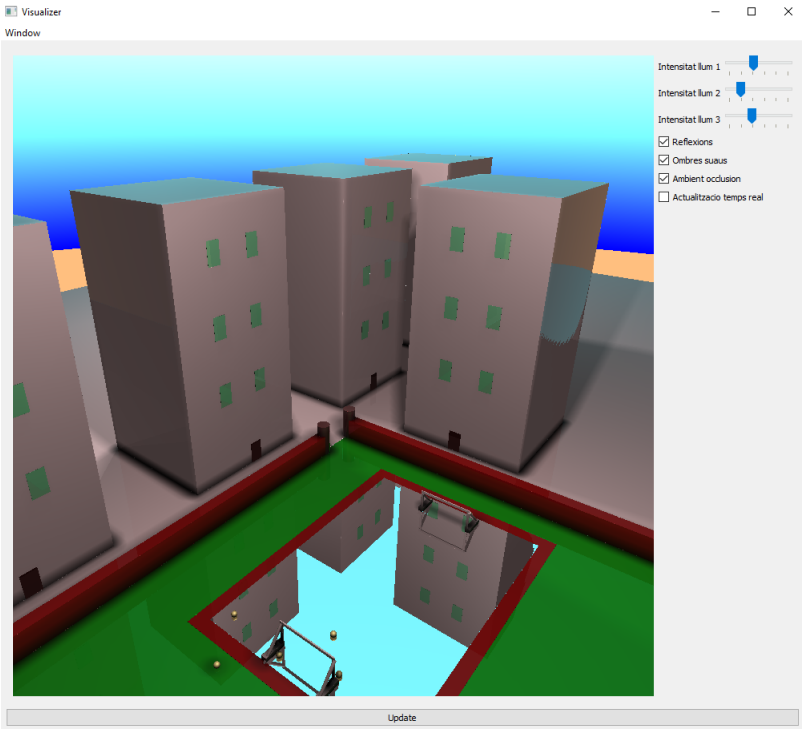


Figura 28: Imatge amb la il·luminació adequada.

5 Anàlisi de les propietats de l'algorisme i l'entorn

L'algorisme utilitzat en el treball és una variant dels algorismes de traçat de rajos amb unes característiques interessants. L'algorisme bàsicament consisteix a traçar un raig per cada píxel en la direcció corresponent segons la posició del píxel, la posició de la càmera i la direcció en la qual apunta la càmera (igual que qualsevol algorisme estàndard de traçat de rajos), però per avançar ho fa mitjançant SDFs (Signed Distance Functions). Les SDFs bàsicament són funcions que contenen informació sobre una o diverses figures geomètriques. Aquestes reben un punt com a paràmetre i retornen la distància mínima entre aquest punt i les figures geomètriques de les quals tenen la informació. L'algorisme aprofita aquestes funcions per a anar avançant la mínima distància que retornen les SDFs i va iterant fins que es troba prou a prop d'una figura o ha arribat a un màxim d'iteracions. La Figura 29 explica gràficament el funcionament del procés explicat i a la referència[13] es poden veure alguns exemples de SDFs.

Un altre dels motius pels quals s'ha decidit utilitzar aquest algorisme és per les característiques del procés de visualització d'OpenGL. En aquest procés cada fragment shader (programa que s'executa per cada píxel) s'executa a la GPU de forma paral·lela. Això es pot aprofitar, ja que en l'algorisme cada raig fa les seves pròpies consultes a les SDFs i pot actuar de forma completament independent.

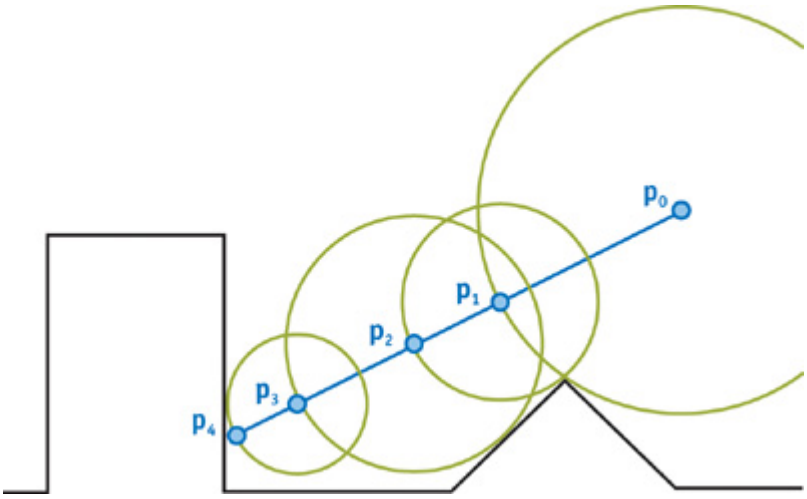


Figura 29: representació gràfica del funcionament de l'algorisme.

5.1 Avantatges

Com s'ha comentat anteriorment en la secció d'estat de l'art, els algorismes de traçat de rajos estan guanyant força i suport. Això és una prova que realment aquests algorismes tenen punts forts, àmbits on poden ser més òptims que la resta i tenen suport dels fabricants de targetes gràfiques i els seus drivers.

També s'ha vist en l'apartat anterior que realment l'algorisme es pot aprofitar completament del paral·lelisme que ofereix OpenGL en la computació del fragment shader.

5.2 Inconvenients

Com s'ha vist en l'apartat anterior, els avantatges que ofereix l'algorisme són molt interessants, però el fet que només es projecti un únic raig per píxel i que aquests rajos no es puguin comunicar entre ells també té inconvenients.

El principal inconvenient que té és que els procediments de suavitzat de canvi de color (com l'antialiàsing per exemple) no són gens eficients d'implementar, ja que aquests procediments acostumen a requerir comunicació entre rajos.

6 Metodologia i rigor

L'àmbit dels gràfics és molt extens i durant els darrers anys s'han descobert i inventat molts procediments i tècniques per tal de millorar el realisme de les imatges processades. El projecte té intenció de poder representar escenes de 3D el màxim realistes. Existeixen diverses tècniques complexes i interessants que poden ajudar a aportar la sensació de realisme. Però tenint en compte que el temps és limitat, caldrà valorar quines tècniques poden aportar més realisme als resultats finals.

En un projecte d'aquestes característiques, s'utilitza una metodologia senzilla i efectiva com la següent: un cop s'ha aconseguit representar l'escena base amb una versió senzilla de l'algorisme, cal escollir una ampliació, implementar-la i finalment realitzar proves i comparacions del funcionament d'aquesta. Un cop aquests passos s'han completat, tornar a començar el procés escollint una nova ampliació.

6.1 Elecció d'una ampliació

Com s'ha esmentat anteriorment, el temps de realització del projecte és limitat i hi ha un nombre considerable d'eines possibles a implementar. Tenint en compte això, cal anar amb cura a l'hora d'escollir en quina millora de l'algorisme es procedirà a treballar a continuació, ja que hi ha opcions més senzilles que d'altres que ofereixen millors resultats finals. També és cert en molts casos que les implementacions que més realisme generen són les que més càlculs requereixen. Veient la situació, la decisió de cada implementació no és trivial.

6.2 Implementació de l'ampliació

Un cop s'ha escollit la següent implementació, s'ha recollit informació sobre el procés algorísmic i s'ha garantit que aquest és assequible d'implementar tenint en compte temps i recursos, es pot iniciar el procés de desenvolupament. Aquest procés pot tenir una durada molt variada segons la complexitat del procés de desenvolupament, la dificultat dels càlculs i els errors que puguin sorgir. Un cop el procés està complet, cal procedir al següent pas.

6.3 Proves i comparacions

Quan sembla que una implementació està acabada, s'ha d'assegurar que realment ho està. Una manera de fer això pot ser generar diferents imatges per comprovar que el resultat és l'esperat. La comprovació del resultat es pot dur a terme mitjançant una comparació amb altres imatges generades mitjançant programes de la competència que utilitzin el mateix procés. Si sembla que el resultat és deficient, cal reconsiderar la implementació i fer un pas enrere cercant possibles errors que hagin causat aquest resultat. En cas que el resultat sigui satisfactori, es pot tornar a començar el procés de selecció d'ampliació per tal de seguir millorant el projecte.

7 Programes i eines externs

7.1 Microsoft Visual Studio

Microsoft Visual Studio[14] és un entorn integrat de desenvolupament (IDE en anglès) que proporciona facilitats a l'hora de programar. Com un editor de text adaptat per a diversos llenguatges de programació, un compilador i una eina per a detectar errors (debugging).

En aquest projecte s'ha utilitzat la versió de Microsoft Visual Studio 2017 amb una ampliació d'aquest per a simular i garantir les compatibilitats que tenia el Visual Studio de 2015. També s'ha afegit un plug-in per a fer-lo compatible amb Qt.

Aquest producte de Microsoft ha facilitat el desenvolupament del projecte en Windows. És cert que en moltes ocasions han sortit errors on faltaven llibreries, arxius o incompatibilitats entre estructures de 32 i 64 bits. Però en projectes on cal comunicar diferents processos i llibreries ja és habitual trobar aquests errors de tant en tant.

Aquest entorn també m'ha permès generar projectes nous il·limitats per a fer proves de compatibilitat entre llibreries, m'ha ajudat a detectar i enfocar errors concrets importants. També té una secció de la pantalla on es poden veure tots els arxius del projecte i permet navegar entre ells amb facilitat. En cas de trobar una funció o classe desconeguda, també permet accedir ràpidament a la declaració, definició o les referències de funcions i classes.

És una eina que en algunes situacions m'ha portat problemes però si hagués de desenvolupar un projecte similar, probablement tornaria a utilitzar el mateix programa.

7.2 Llibraries d'OpenGL

OpenGL és el sistema de renderització que s'ha utilitzat per a desenvolupar aquest projecte. Inicialment s'han utilitzat les llibreries de "freeglut" i "glew" per a programar i utilitzar els elements d'OpenGL.

El procés de renderització d'OpenGL conté diversos procediments que es resumiran a continuació.

7.2.1 Especificació dels vèrtexs

Aquest procés consisteix en la reunió de la suficient informació sobre tots els vèrtexs que hi haurà a l'escena per a poder cridar al vertex shader. S'omplen diverses estructures de dades amb les coordenades de cadascun dels vèrtexs, com el Vertex Array Object (VAO) i el Vertex Buffer Object (VBO). Un cop està la informació preparada, es procedeix al següent pas.

7.2.2 Processament dels vèrtexs

Aquest pas també s'anomena vertex shader. És un pas que permet moltes possibilitats perquè el codi que hi ha en el fragment shader s'executa per cadascun dels vèrtexs independentment. Aquest procés té alguns passos opcionals, com el "tessellation" i el "geometry shader".

7.2.3 Postprocessament dels vèrtexs

En aquest procés s'executen diverses operacions sobre els vèrtexs utilitzant qualsevol informació addicional que s'hagi pogut afegir en el vertex shader per a preparar el següent pas.

7.2.4 Reunió de primitives i rasterització

Després de tractar un seguit de vèrtexs es generen el que s'anomena primitives. Un cop el pas anterior acaba la seva feina, les primitives es divideixen en seqüències de primitives individuals i es crida al procés de rasterització. En aquest procés se separen les primitives per a generar el que s'anomena fragments. Un fragment bàsicament correspon a un píxel.

7.2.5 Processament de fragments

Aquest procés té un concepte similar al del vertex shader. Aquí es tracta cada fragment individualment. Aquest pas d'OpenGL és el que més pes ha tingut a l'hora de desenvolupar el projecte.

7.2.6 Tractament final dels fragments

Finalment es fan un seguit de tests i processats per a determinar el resultat final de l'escena.

7.3 Qt

Qt és un equip de desenvolupament de programari (SDK en anglès) que permet el disseny i implementació d'interfícies de manera senzilla. En aquest projecte ha resultat ser una eina molt útil i interessant per a facilitar considerablement la manipulació de l'escena. Tant en facilitat com en intuïtivitat.

Qt també consta de llibreries pròpies d'OpenGL que si s'utilitzen, asseguren la compatibilitat entre les dues eines.

7.4 GitHub

GitHub és una plataforma de desenvolupament que permet mantenir un control de versions de projectes al núvol de manera gratuïta. Aquesta eina s'ha utilitzat per a mantenir tots els projectes de forma segura i poder treballar des de diferents llocs.

7.5 Lleis i regulacions

A l'hora d'utilitzar programes i llibreries externes cal anar amb cura i tenir en compte que aquests programes i llibreries potser no estan pensats per a programes d'aquestes característiques i no són gratuïts. Considerant això, s'ha fet una revisió de les i programes utilitzats en el projecte on s'ha vist que el projecte no està gaire limitat per lleis o normatives. Les úniques que poden afectar són les llicències dels programes i llibreries utilitzats per a desenvolupar aquest. A continuació s'analitzen les normatives de cadascuna de les dependències utilitzades.

7.5.1 Microsoft Visual Studio

Per una banda s'ha utilitzat Microsoft Visual Studio 2017, que com a alumne de la UPC disposo d'una llicència oficial per a utilitzar aquest.

7.5.2 OpenGL

Per altra banda també s'han utilitzat llibreries d'OpenGL. Com es pot veure a la referència[10], tot el codi basat en l'API d'OpenGL no tenen cap requisit de llicència.

7.5.3 GitHub

Per a garantir la seguretat del projecte cap a qualsevol tipus de problemes s'ha utilitzat GitHub, on es pot veure en la referència[4] que serveis de manera gratuïta.

7.5.4 Qt

I finalment s'ha utilitzat l'eina Qt. En el cas que aquesta s'utilitzi, també cal tenir en compte les llicències d'aquesta. Qt disposa de 2 llicències[11]. Una gratuïta i una de pagament. En el cas del projecte desenvolupat es compleixen les característiques per a poder utilitzar la Qt de forma gratuïta.

8 Plà de desenvolupament

El desenvolupament d'un motor gràfic que pugui representar escenes de qualitat requereix un bon algorisme amb un nombre considerable d'ampliacions visuals. I com s'ha esmentat anteriorment, el temps de desenvolupament és limitat. Això implica que cal escollir amb cura les ampliacions a desenvolupar i és important tenir una referència temporal per controlar que les tasques més importants estaran aplicades i funcionaran correctament a la data d'entrega. També cal tenir en compte que hi ha diferents tipus de feines a fer: generar documentació, ampliacions de l'algorisme i ampliacions d'usabilitat. I cadascuna d'aquestes tasques s'ha de realitzar correctament. A continuació s'analitzarà el temps disponible de cada fase de desenvolupament i s'enumeraran les tasques més rellevants de cada tipus i s'estimarà el cost temporal d'aquestes.

Es disposa de 4 mesos aproximadament per a realitzar el projecte sencer, i això implica per una banda desenvolupar un programa complet i competent i per altra banda generar una bona documentació sobre totes les característiques i detalls del programa. També cal tenir en compte que la documentació està separada en diverses parts i cadascuna d'aquestes té uns requeriments, unes limitacions i una data d'entrega. Pràcticament tota la generació de la documentació forma part del que s'anomena "fita inicial", que consisteix en un seguit de lliuraments que contenen diferents parts de la documentació del projecte.

Un cop els lliuraments que formen part de la fita inicial s'han entregat comença el període per a la "fita final", on s'haurà de presentar el desenvolupament real del projecte amb una memòria. Per a poder tenir control del desenvolupament i disposar d'una referència temporal durant el mateix desenvolupament, cal dividir el procés en diverses parts i planificar cadascuna amb una estimació del cost temporal.

8.1 Fita inicial

La fita inicial està dividida en 6 lliurables de documentació i conté una presentació oral al final d'aquesta:

- Lliurable 1 "Abast del projecte i contextualització".
- Lliurable 2 "Planificació temporal".
- Lliurable 3 "Gestió Econòmica i sostenibilitat".
- Lliurable 4 "Presentació preliminar".
- Lliurable 5 "Plec de condicions".
- Lliurable 6 "Presentació oral i document final".

8.2 Fita final

Un cop la fita inicial està acabada, es disposa d'una base sòlida a partir de la qual es pot generar una bona documentació per a suportar el projecte final. En aquest precís moment també es disposa d'un augment de temps disponible per procedir al desenvolupament del motor gràfic en si. Aquest període de temps per al desenvolupament del projecte acaba amb la fita final, on s'ha de presentar el projecte final amb la memòria definitiva.

8.3 Descripció de les tasques

Un cop es té constància de les fites i les prioritats de cadascuna, és més senzill planificar tot el projecte en tasques i decidir quina tasca és més important en cada moment. Tenir aquesta planificació també ajuda a l'hora de treballar, ja que l'única preocupació durant el projecte és seguir cadascuna de les tasques que es troba en la planificació sense necessitat de fer-se les preguntes "quina tasca caldria fer a continuació?" i "és realment la millor idea fer la tasca ara?".

Veient la importància de tenir una planificació, també cal tenir en compte que aquesta planificació ha d'estar ben pensada. En el cas que la planificació no fos prou bona i tingués errors, possiblement a mig projecte es pot detectar l'error i això implicaria fer un pas enrere, replanificar el projecte i seguir treballant. I aquest pas enrere seria una pèrdua de temps i d'esforç que es pot evitar fàcilment si es fa una bona feina a l'hora de planificar. A continuació es presenta la planificació que s'ha establert per a aquest projecte tenint en compte les fites i el temps disponible:

- Generació de documentació complint les dates dels lliuraments de la fita inicial: redacció de la part principal de la memòria.
- Configuració del projecte en Windows: el projecte es desenvoluparà en Windows. Per tant, és el primer sistema operatiu on ha de funcionar.
- Generació d'escenes bàsiques: un cop el projecte està configurat adequadament, cal preparar el projecte base sobre el qual es desenvoluparan totes les ampliacions posteriors.
- Adaptació del projecte per a Linux: és important que el projecte sigui compatible amb diversos sistemes operatius. Això augmentarà molt les possibilitats del projecte.
- Implementació del càlcul de llum: no és una ampliació senzilla, però és la base del realisme. A més, permet moltes més possibilitats a l'hora de generar escenes.
- Implementacions visuals addicionals: un cop les tasques anteriors estan finalitzades, es poden començar a implementar detalls extra de visualització per tal de millorar els resultats.
- Generació de la memòria final i presentació del projecte per a complir la fita final: amb el projecte acabat cal preparar la documentació final i la presentació.

8.4 Dependències entre tasques

Un cop les tasques estan definides, cal especificar les dependències entre elles. Hi ha algunes tasques que es poden fer simultàniament i d'altres que necessiten que certa tasca estigui finalitzada anteriorment.

En el cas de la documentació, aquest procediment és completament independent del desenvolupament del projecte.

La configuració del projecte per a Windows és essencial que sigui la primera tasca a finalitzar perquè és l'entorn on s'ha desenvolupat el projecte.

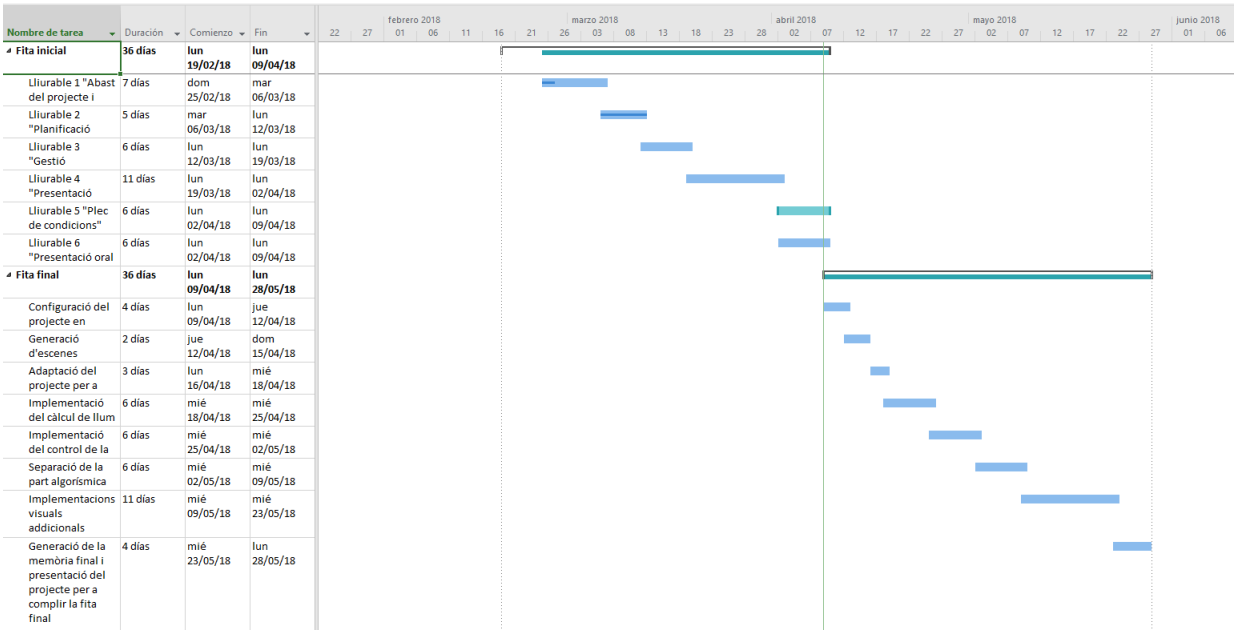
La generació de les escenes més bàsiques és també imprescindible per a seguir avançant amb el projecte.

L'adaptació per a Linux és una adaptació independent que es pot fer en qualsevol moment del projecte. Ferla just després de la configuració a Windows i la generació de les escenes bàsiques és el moment més idoni, però es pot fer més endavant sense cap problema perquè no té cap efecte en el codi i l'algorisme.

A continuació les millores del càlcul de llum, el control de la càmera, la separació entre la part algorísmica i el codi i les millores visuals addicionals no tenen necessitat de cap ordre concret.

8.5 Diagrama de GANTT

A continuació es mostra el diagrama de la planificació del projecte:



Nombre de tarea	Duración	Comienzo	Fin
Fita inicial	**36 días**	**lun 19/02/18**	**lun 09/04/18**
Lliurable 1 "Abast del projecte i	7 días	dom 25/02/18	mar 06/03/18
Lliurable 2 "Planificació	5 días	mar 06/03/18	lun 12/03/18
Lliurable 3 "Gestió	6 días	lun 12/03/18	lun 19/03/18
Lliurable 4 "Presentació	11 días	lun 19/03/18	lun 02/04/18
Lliurable 5 "Plec de condicions"	6 días	lun 02/04/18	lun 09/04/18
Lliurable 6 "Presentació oral	6 días	lun 02/04/18	lun 09/04/18
Fita final	**36 días**	**lun 09/04/18**	**lun 28/05/18**
Configuració del projecte en	4 días	lun 09/04/18	jue 12/04/18
Generació d'escenes	2 días	jue 12/04/18	dom 15/04/18
Adaptació del projecte per a	3 días	lun 16/04/18	mié 18/04/18
Implementació del càlcul de llum	6 días	mié 18/04/18	mié 25/04/18
Implementació del control de la	6 días	mié 25/04/18	mié 02/05/18
Separació de la part algorísmica	6 días	mié 02/05/18	mié 09/05/18
Implementacions visuals addicionals	11 días	mié 09/05/18	mié 23/05/18
Generació de la memòria final i presentació del projecte per a complir la fita final	4 días	mié 23/05/18	lun 28/05/18

9 Valoració d'alternatives i pla d'acció

Anteriorment s'ha esmentat que és molt important tenir una bona planificació per evitar al màxim la situació on cal revisitar-la perquè alguna data no s'ha pogut complir o algun pas en la planificació no ha estat prou rumiat. Però a la pràctica és molt habitual que sorgeixin contratemps i molts cops és pràcticament impossible tenir una planificació prou bona per a evitar l'aplicació d'alternatives.

Tenint això en compte, cal tenir un pla d'acció en cas que el projecte es trobi en situació on la planificació no es pot seguir al peu de la lletra. També s'ha de considerar que això és molt habitual en la informàtica. No és estrany que els processos (tant de desenvolupament com de configuració) es compliquin per qualsevol petit error fàcil d'arreglar però difícil de detectar. Això fa que sigui completament impossible de determinar exactament la durada en la qual s'incrementaria el temps de desenvolupament si aparegués un d'aquests problemes. Una estimació raonable podria ser la pèrdua d'1 dia de desenvolupament per cadascun d'aquests errors que no sigui trivial de detectar/arreglar.

També cal tenir en compte que és possible que qualsevol dels components de l'ordinador en el qual es desenvolupa el projecte falli. En aquest cas cal adquirir un component nou. El procés d'adquirir un component nou és d'1 dia laborable sencer.

Després de considerar aquestes possibilitats, la planificació té la propietat que les tasques estan ordenades de més important a menys important. Això implica que si s'allarga el procés de desenvolupament de qualsevol de les tasques, és prioritari acabar la tasca anterior abans de seguir. En cas de trobar-se en aquesta situació, l'únic que cal fer a la planificació és allargar terminis. No cal afegir, esborrar o modificar les tasques establertes.

Amb aquest plantejament i pla d'acció s'espera evitar al màxim trobar-se en la situació de necessitar fer un pas enrere, deixar de treballar en el que s'està desenvolupant i tractar tasques que anterior s'hagin deixat a mig fer.

10 Autoavaluació de la sostenibilitat

Després de realitzar l'enquesta sobre els coneixements de sostenibilitat social, econòmica i ambiental es pot veure que el fet d'haver assistit a diverses xerrades durant els darrers anys realment ajuda a conscienciar als assistents sobre l'impacte que pot tenir en el món cadascuna de les decisions sobre el projecte. A l'hora de prendre qualsevol tipus de decisió és més fàcil adonar-se i preveure l'impacte que pot tenir socialment, econòmicament i ambientalment.

10.1 Consciència social

Després d'haver adquirit alguns coneixements sobre sostenibilitat, es té en compte que qualsevol decisió en un projecte relacionat amb TIC pot afectar a la societat de diverses formes. Per una banda, els projectes amb intenció d'accedir al mercat sempre tenen intenció de facilitar o ajudar en algun procés de qualsevol tipus i causar una bona impressió als usuaris. Però això pot implicar empitjorar la situació d'algun altre col·lectiu. Per exemple, el nou programa desenvolupat per a certa empresa permet facilitar el procés i, per tant, es pot reduir la plantilla de treballadors. En l'exemple plantejat, el programa té un bon impacte en un grup social, però un impacte bastant negatiu en els treballadors acomiadats.

Aquest tipus de problemes també poden aparèixer en els projectes que impliquen la producció d'aparells electrònics, ja que en aquest cas s'estarà donant suport a la mineria del mineral coltan[16] i el procés de mineria d'aquest mineral està causant patiment a milers de persones.

10.2 Consciència ambiental

Un projecte molt ambiciós pot tenir unes conseqüències ambientals molt dolentes si no es té en compte i no s'intenta fer res al respecte. Un exemple seria el cas on el projecte consisteix a renovar una peça concreta de tots els ordinadors sense reutilitzar l'antiga. Aquest projecte té potencial de generar moltes deixalles, i si no es va amb compte a l'hora de llançar aquestes deixalles, poden contribuir a l'augment de la contaminació i a empitjorar la qualitat de vida dels animals i de les persones que viuen prop de les zones on es llancen les deixalles electròniques.

10.3 Consciència econòmica

L'aspecte econòmic d'un projecte habitualment és el prioritari. Per una banda això és bo, ja que el projecte serà el millor possible, el màxim competitiu possible i intentarà ajudar a la societat de la millor manera possible. Però per altra banda això pot ser el motiu pel qual un projecte intenta obtenir el màxim de beneficis sense tenir en compte les conseqüències socials i ambientals.

11 Dimensió econòmica

Arribats al punt on s'ha decidit desenvolupar el projecte i es té en compte que aquest pot tenir impacte social, econòmic i ambiental, l'únic que pot impedir el desenvolupament d'aquest és la mancança de recursos econòmics. Per això, cal tenir constància de totes les possibles despeses i tots els possibles beneficis. També és molt important considerar imprevistos, ja que si no es té en compte, un imprevist prou gran pot fins i tot forçar una cancel·lació del desenvolupament del projecte. A continuació s'estimaran els costos del projecte amb les seves característiques.

11.1 Identificació dels costos

Els costos d'aquest projecte es poden distribuir en recursos humans, hardware, software, despeses generals (indirectes) i llicències. A continuació es fa una anàlisi de cadascun dels tipus de despeses.

11.1.1 Recursos humans

Pel que fa als recursos humans, s'han definit 4 rols necessaris per a l'elaboració del projecte: cap de projecte, dissenyador, desenvolupador i tester.

El cap del projecte és l'encarregat que el projecte segueixi endavant i que es compleixin les dates de finalització de cada procés en mesura del possible. És important que tingui molt bones habilitats de comunicació, que tingui experiència en projectes similars i que conegui a la perfecció la feina que ha de realitzar cadascun dels treballadors dels quals s'encarrega.

El dissenyador és qui decideix com estarà fet el projecte i com s'hauria de veure el resultat final. Això implica decidir com serà la interfície, decidir quines estructures de dades s'utilitzaran i com s'utilitzaran. Aquest membre també ha de tenir experiència en projectes similars i ha de conèixer molt bé el rol de desenvolupador.

El desenvolupador és qui segueix les instruccions del dissenyador i qui fa la programació necessària per al projecte. Aquest membre necessita coneixements avançats de programació en C++[2] i OpenGL.

El tester és qui comprova exhaustivament totes les funcionalitats del projecte i qui assegura que el projecte no té cap error inesperat. Aquest membre no necessita gaire formació.

Tenint en compte que dels 72 dies que s'especifiquen al GANTT es treballen 5 h al dia, el total d'hores és de 360.

Recursos Humans			
Rol	Hores	Preu/hora	Preu total
Cap de projecte	180	50€/h	9.000€
Dissenyador	50	40€/h	2.000€
Desenvolupador	100	35€/h	3.500€
Tester	30	30€/h	900€
TOTAL	360		15.400€

11.1.2 hardware

Els components necessaris per a dur a terme el desenvolupament d'aquest projecte no són massa costosos ni abundants. En aquest projecte es necessitarà un ordinador de sobretaula amb una targeta gràfica i un processador decents. El cost total d'adquirir un ordinador amb capacitat de dur a terme el desenvolupament és de 600€.

A part de la targeta gràfica i el processador, caldrà una font d'alimentació, una placa base, un disc dur, 8gb de memòria RAM, una torre per a unir totes les peces, una pantalla, un ratolí i un teclat. Com que el cost d'adquirir aquestes peces no és massa elevat, a la taula es consideren com a "resta d'elements".

Hardware				
Producte	Preu	unitats	Vida útil	Amortització
Targeta gràfica	250€	1	4 anys	12€
Processador	150€	1	4 anys	7€
Resta d'elements	200€	1	4 anys	10€
TOTAL	600€			29€

11.1.3 software i llicències

Per a desenvolupar tant la documentació com el codi, s'ha utilitzat diferents programes, algun d'aquests necessita llicència per a ser utilitzat professionalment. Aquests programes són: Microsoft Windows 10 com a sistema operatiu on es desenvoluparà el projecte, Microsoft Visual Studio 2015 com a editor i la llicència de Microsoft Project per a gestionar el projecte.

Software				
Producte	Preu	unitats	Vida útil	Amortització
Microsoft Windows 10	199€	1	3 anys	13€
Microsoft Visual Studio	499€	1	3 anys	32€
Microsoft Project	30€/mes	1	3 anys	75€
TOTAL	698€ + 30/mes			120€

11.1.4 despeses generals

Les despeses generals o indirectes són inevitables i també s'han de tenir en compte. Aquestes despeses surten de 4 factors: lloguer del local de treball, aigua, electricitat i paper.

Generals			
Producte	Preu	unitats	Aproximació del cost
Lloguer del local	100€/mes	2.5 mesos	250€
Aigua	30€/mes	2.5 mesos	75€
Electricitat	0,07€/kWh	70.000 kWh	4.900€
Paper	30€/pila	1 pila	30€
TOTAL	2194,80€/any		5.255€

11.2 Estimació dels costos totals i imprevistos

Com s'ha pogut veure anteriorment, hi ha diverses fonts de gastos que s'han de cobrir si es vol desenvolupar un projecte en condicions. A continuació hi ha una taula amb el resum total dels costos tenint en compte que ampliarem el pressupost afegint un 20

Costos	
Font	Preu
Recursos humans	22.150€
Hardware	600€
Software	878€
General	5.255€
Imprevistos	5.776,60€
TOTAL	34.659,6€

11.3 Control de gestió

Per a justificar els imprevistos cal intentar preveure quins són aquests possibles imprevistos, les seves possibilitats i el que podrien arribar a augmentar el cost de desenvolupament. Per això s'ha creat la següent taula amb els possibles imprevistos i les seves característiques:

Contingències			
Risc	Probabilitat	Cost estimat	Exposició al risc
Reparació total ordinador	40%	500€	200€
Allargada desenvolupament	50%	10.000€	5.000€
Baixa treballador	10%	5.000€	500€
TOTAL			5.700€

12 Reflexió

El motor gràfic en el qual s'està treballant també pot tenir impacte econòmic, ambiental i social a la societat. A continuació es reflexionarà sobre el possible impacte que pot tenir el projecte en cadascun dels àmbits.

12.1 Econòmica

Respecte a les condicions econòmiques, el projecte està desenvolupat per un grup reduït de persones i durant un període curt de temps. Això fa que el projecte no necessiti una gran inversió per a ser desenvolupat. També s'ha de tenir en compte que aquest projecte té intenció d'accedir a un mercat molt potent, pel que li dóna molt de potencial a generar molts beneficis si s'aconsegueix establir a les empreses. També cal tenir en compte que l'algorisme utilitza una tècnica eficient que s'està posant de moda recentment, el que fa que la competència difícilment té bones implementacions del mateix algorisme. Una altra propietat de l'algorisme és que s'aprofita de la potència de les targetes gràfiques, i les targetes gràfiques són un element del mercat que té molta competència i està en contínua millora.

12.2 Ambiental

El projecte consisteix completament en software, el que implica que no millorarà cap element de hardware antic ni forçarà la generació de cap tipus de deixalles ni contribuirà directament a l'extracció de coltan. És cert per això que per a visualitzar bones escenes fa falta una targeta gràfica decent, i això sí que pot tenir algun efecte negatiu al medi ambient si l'usuari necessita renovar la targeta gràfica. Però aquest últim inconvenient és inevitable si la intenció és fer un producte competitiu. Però com que actualment tots els programes de gràfics necessiten bones targetes gràfiques, difícilment un client del projecte necessitarà renovar el material.

12.3 Social

Pel que fa a l'efecte social, el desenvolupament d'un projecte d'aquestes característiques aporta una experiència molt interessant en un àmbit creixent de la informàtica. Respecte a la competència, com que el projecte té intenció de proposar alternatives als altres programes de visualització, aquest pot utilitzar elements de disseny millorats respecte als programes existents. El projecte desenvolupat tampoc és un programa absolutament necessari en l'actualitat, hi ha diverses alternatives. Però aquesta situació és un indicador que una bona eina pot imposar-se a la competència.

Referències

[1] Game Developers Conference. *Conferència anual on els desenvolupadors de videojocs més rellevants es reuneixen per a compartir idees.* URL: http://www.gdconf.com/.

[2] cplusplus.com. *Llenguatge de programació.* URL: http://www.cplusplus.com/.

[3] Igor Dykhta. *article sobre càlcul de components d'il·luminació.* URL: http://sunandblackcat.com/tipFullView.php?l=eng&topicid=30&topic=Phong-Lighting.

[4] GitHub. *Informació legal de GitHub.* URL: https://help.github.com/articles/github-terms-of-service/.

[5] Naty Hoffman. *Presentació efecte Fresnel.* URL: http://blog.selfshadow.com/publications/s2015-shading-course/hoffman/s2015_pbs_physics_math_slides.pdf.

[6] Benjamin Keinert et al. "Enhanced Sphere Tracing". In: *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*. Ed. by Andrea Giachetti. The Eurographics Association, 2014. ISBN: 978-3-905674-72-9. DOI: 10.2312/stag.20141233.

[7] Etay Meiri. *Tutorial OpenGL.* URL: http://ogldev.atspace.co.uk/.

[8] NVIDIA. *Conferència recent NVIDIA.* URL: https://www.ustream.tv/gpu-technology-conference.

[9] NVIDIA. *Web oficial de l'empresa NVIDIA.* URL: http://www.nvidia.es/page/home.html.

[10] OpenGL. *Informació sobre OpenGL, a la part inferior de la web es pot veure la secció 'Licensing' on parla de les característiques le la llicència.* URL: https://www.opengl.org/about/#11.

[11] Qt. *Informació sobre les 2 llicències que ofereix Qt.* URL: https://www1.qt.io/licensing/.

[12] Qt. *Tutorial OpenGL i Qt base.* URL: http://doc.qt.io/qt-5/qtopengl-hellogl2-example.html.

[13] Inigo Quiulez. *Signed Distance Functions.* URL: http://iquilezles.org/www/articles/distfunctions/distfunctions.htm.

[14] Microsoft Visual Studio. *IDE de microsoft.* URL: https://www.visualstudio.com/es/.

[15] Wikipedia. *article en anglès sobre la il·luminació de 3 punts.* URL: https://en.wikipedia.org/wiki/Three-point_lighting.

[16] Wikipedia. *Coltan, mineral utilitzat en la producció d'aparells electrònics.* URL: https://ca.wikipedia.org/wiki/Coltan.

[17] Wikipedia. *OpenGL.* URL: https://en.wikipedia.org/wiki/OpenGL.

[18] Wikipedia. *QT, llibreria que permet que el programa pugui llegir el moviment del ratolí o la pressió d'una tecla del teclat.* URL: https://en.wikipedia.org/wiki/Qt_(software).